Ma YL, Yuan Y, Zhu FD *et al.* Who should be invited to my party: A size-constrained k-core problem in social networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(1): 170–184 Jan. 2019. DOI 10.1007/s11390-019-1905-0

Who Should Be Invited to My Party: A Size-Constrained k-Core Problem in Social Networks

Yu-Liang Ma[1], *Member*, *CCF*, Ye Yuan[1], *Member*, *CCF*, *ACM*, *IEEE*, Fei-Da Zhu[2], *Member*, *ACM*, *IEEE*
Guo-Ren Wang[3], *Member*, *CCF*, *ACM*, *IEEE*, Jing Xiao[4], and Jian-Zong Wang[4], *Member*, *CCF*

[1] *School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China*
[2] *School of Information Systems, Singapore Management University, Singapore 188065, Singapore*
[3] *School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*
[4] *Ping An Technology (Shenzhen) Co., Ltd, Shenzhen 518048, China*

E-mail: ylma.neuer@gmail.com; yuanye@mail.neu.edu.cn; fdzhu@smu.edu.sg; wanggrbit@126.com
{xiaojing661, wangjianzong347}@pingan.com.cn

Received November 28, 2017; revised September 20, 2018.

Abstract In this paper, we investigate the problem of a size-constrained k-core group query (SCCGQ) in social networks, taking both user closeness and network topology into consideration. More specifically, SCCGQ intends to find a group of h users that has the highest social closeness while being a k-core. SCCGQ can be widely applied to event planning, task assignment, social analysis, and many other fields. In contrast to existing work on the k-core detection problem, which aims to find a k-core in a social network, SCCGQ not only focuses on k-core detection but also takes size constraints into consideration. Although the conventional k-core detection problem can be solved in linear time, SCCGQ has a higher complexity. To solve the problem of SCCGQ, we propose a Blast Scatter (BS) algorithm, which appoints the query node as the center to begin outward expansions via breadth search. In each outward expansion, BS finds a new center through a greedy strategy and then selects multiple neighbors of the center. To speed up the BS algorithm, we propose an advanced search algorithm, called Bounded Extension (BE). Specifically, BE combines an effective social distance pruning strategy and a tight upper bound of social closeness to prune the search space considerably. In addition, we propose an offline social-aware index to accelerate the query processing. Finally, our experimental results demonstrate the efficiency and effectiveness of our proposed algorithms on large real-world social networks.

Keywords group query, k-core, social analysis, social network

1 Introduction

The emergence of online social networks, which has become an indispensable part of people's lives, poses significant research challenges for social network analysis. One of the fundamental frameworks used to analyze social networks is the k-core model, and in this paper, we propose a novel size-constrained k-core group query (SCCGQ) problem based on this model. Specifically, given an underlying social network $G = (V, E)$, a query user $u_q \in V$, a query group size constraint h, and a core number k, SCCGQ aims to find a user group with size h (including the query user u_q) that constitutes a k-core structure. That is, the induced subgraph of the users in the h-size group is a connected graph, and the degree, in the induced subgraph, of each user (the number of induced subgraph edges connecting to the user) is no less than k. The user group returned by SCCGQ should have the highest closeness among all the possible groups satisfying the above conditions. SCCGQ has a

Regular Paper

This research is partially supported by the National Research Foundation, Prime Ministers Office, Singapore, under its International Research Centres in Singapore Funding Initiative and Pinnacle Lab for Analytics at Singapore Management University, the National Natural Science Foundation of China under Grant Nos. 61572119, 61622202, 61732003, 61729201, 61702086, and U1401256, and the Fundamental Research Funds for the Central Universities of China under Grant No. N150402005.

wide range of applications, such as event planning[1−4], task scheduling[5−9], and social analysis[10,11]. In the following, we give two concrete application examples.

Scenario 1: *Event Planning.* Assume that Alice wants to hold a dinner party for 10 people. Deciding who should be invited is an annoying problem. Due to the limited space, this party allows a maximum of 10 participants. The guests may feel isolated if they are strangers to one another. Alice expects that each participant knows at least three of the others, and the group of 10 should be as close as possible. Thus, this scenario can be modeled as an SCCGQ problem, which will return a 3-core user group with size 10, including Alice. This group should have the highest closeness among all feasible groups.

Scenario 2: *Task Scheduling.* Suppose Bob is a contractor who has a new project to complete. Bob wants to find a group of workers to accomplish this project together. Due to the limitations of project funding as well as the deadline of the project, the group size is set to 9 (including Bob). To guarantee good cooperation among the project team, Bob expects that each member knows at least three of the others, and the group closeness should be as high as possible. Therefore, this scenario can be modeled as an SCCGQ problem as well, which will return a user group with size 9 and the highest social closeness while being a 3-core.

In this paper, we propose the SCCGQ problem not only for its importance in real life but also for the significance of the k-core structure in social network analysis. The definition of "k-core" is that each node connects to at least k other nodes, rather than to all other nodes in the structure (clique structure). The k-core structure relaxes social topology constraints and makes the SCCGQ results more diverse. Additionally, finding a k-core structure provides an opportunity to expand the social circle of each user in the core structure. Taking scenario 1 as an example, Alice invited nine people to her party, which provided an opportunity for each participant to become acquainted with new friends and promoted the development of social relationships.

It is worth noting that the SCCGQ problem is different from the conventional k-core problem, which aims to find a maximally connected subgraph with all vertices having a degree of at least k. The main differences are summarized as follows. 1) The SCCGQ problem has a size constraint. The solution to the conventional k-core detection problem is very likely to exceed the size constraint of our SCCGQ problem and thus is not an answer to our proposed problem here. 2) The SCCGQ problem takes social closeness into consideration, which makes it more interesting and meaningful. The k-core involving the query user and satisfying the size constraint may not be unique in most cases. Among these k-cores, the one with the highest closeness is the answer to our query, as a higher social closeness can benefit the development of social relationships. It is worth noting that the group size constraint and the social closeness are both important and meaningful in real-life applications.

A naive solution to SCCGQ can be readily described as follows. First, via a k-core decomposition algorithm, we obtain a node set, denoted by G_k, in which the core number of each node is no less than k. Second, we examine every subset of G_k with size h. These subsets, whose induced graphs constitute k-core structures and include the query node, can be seen as candidates. Finally, the candidate with the highest closeness is returned as the SCCGQ result. The time complexity of the naive solution is $O(n^h \times h^2)$, where n is the number of total nodes in the underlying social network. The time taken to select h nodes is $O(n^h)$, and the time taken to estimate whether the induced subgraph of a set of h nodes constitutes a k-core structure is $O(h^2)$. When h is a constant, SCCGQ can be solved in polynomial time, while SCCGQ has exponential complexity when h is a part of the input[12]. In this paper, we focus on the latter, that is, parameter h is a part of the input.

Apparently, the naive solution is extremely time consuming and inefficient. To address the SCCGQ problem, we propose a Blast Scatter (BS) algorithm, which appoints the query node as the center to start outward expansions via a breadth search. In each outward expansion, BS first selects a new center through a greedy strategy and then selects multiple neighbors of the center to guarantee that the number of nodes connecting to the center in the current expansion is no less than k and that the size of the group does not exceed h. When the size of an expansion reaches h, BS checks whether the induced subgraph of the h nodes constitutes a k-core structure. Finally, after all possible expansions have finished, the induced subgraph with the highest closeness is returned as the result.

Moreover, to speed up the BS algorithm, we propose an advanced search algorithm, called Bounded Extension (BE). Specifically, BE combines an effective social distance pruning strategy and a tight upper bound of social closeness to prune the search space considerably. The social distance pruning strategy takes both social

graph topology and the query group size constraint into consideration. Meanwhile, for a possible result P, the number of edges in the induced subgraph of P can be bounded, as the size of P is constrained by h. Thus, a tight upper bound of social closeness with respect to a user group can be deduced.

Contributions. Our major contributions in this work are summarized as follows.

• We are the first to propose the size-constrained k-core group query (SCCGQ) problem for social networks.

• We propose a novel algorithm, Blast Scatter (BS), which appoints the query node as a center to start outward expansions via a breadth search and selects multiple neighbors of the center at a time to accelerate the query process.

• To speed up the BS algorithm, we design an advanced search algorithm, called Bounded Extension (BE) that combines an effective social distance pruning strategy and a tight upper bound of social closeness to prune the search space considerably. In addition, we propose an offline social-aware index to accelerate the query processing.

• Extensive experiments are conducted to demonstrate the efficiency and effectiveness of our proposed algorithms on large real-world social networks.

The rest of this paper is organized as follows. We formally define the SCCGQ problem in Section 2. In Section 3, we give an elaboration of our Blast Scatter (BS) algorithm, describe the social distance pruning strategy, and present our Bounded Extension (BE) algorithm. We present experimental evaluations in Section 4, and provide an overview of the related work in Section 5. We conclude this paper with some discussions in Section 6.

2 Preliminaries and Problem Formulation

In this section, we describe the terms and notations that we use throughout the paper and then formally define the problem of size-constrained k-core group query (SCCGQ).

Let $G = (V, E, W)$ be an undirected weighted social graph, where V is the set of nodes representing users in the social network and E is the set of edges in G denoting social ties. Each edge $e(u, v) \in E$ is associated with a weight $\omega \in W$ indicating the closeness between the two users. Currently, there are many measures to estimate the closeness between two users, such as the cosine similarity[13], the Jaccard coefficient[14], and random walk[15]. In this paper, we adopt the Jaccard coefficient, which has been widely used to measure the weight between two users. It is worth noting that our method can be easily extended to other measures. The weight between user u and its neighbor v, denoted as $\omega(u, v)$, is the Jaccard coefficient between their neighbor sets $N(u)$ and $N(v)$, where $N(x) = \{n|(x, n) \in E\}$. Therefore, for $(u, v) \in E$, we have:

$$\omega(u, v) = \frac{|N(u) \cap N(v)| + \theta}{|N(u) \cup N(v)|}. \tag{1}$$

The numerator of (1) is θ plus the intersection of the neighbor sets of two users, where θ is a positive constant. In this paper, we set the value of θ as the average number of neighbors. In this way, we can guarantee that the closeness of two users is nonzero if there exists an edge connecting them and the intersection of their neighbor sets is empty.

The closeness of a connected graph $G = (V, E)$ is the sum of all the edge weights in G, denoted as $Co(G) = \sum_{e \in E} \omega(e)$. Significantly, it is meaningless to discuss the closeness of an unconnected graph, for instance, when there is no connection between two user groups, which can be modeled as an unconnected graph $G = (V_1, V_2, E_1, E_2)$. Discussing the closeness of each of the two user groups is meaningful while discussing the overall closeness of the unconnected graph is meaningless, as the two user sets are separated and independent in a sense.

Definition 1 (Induced Subgraph). *Given a graph $G = (V, E)$, for any subset V' of V and edge set E':*

$$E' = \{(u, v)|u, v \in V' \text{ and } (u, v) \in E\},$$

we call $G' = (V', E')$ an induced subgraph of V' in G, denoted as $G[V']$.

The concept of a k-core was first proposed by Seidman in [16]. It can be widely applied to describe the complex topology of social networks and reveal the hierarchical structures of networks.

Definition 2 (k-Core). *A k-core of a graph G is a maximal connected subgraph of G in which all nodes have a degree of at least k.*

Definition 3 (Coreness). *If a node v belongs to a k-core but does not belong to any $(k + 1)$-core, we call k the coreness of v.*

Definition 4 (Size-Constrained k-Core). *Given a social graph $G = (V, E)$, a size constraint h, and a node set $C \subseteq V$, we call an induced subgraph $G[C]$ a size-constrained k-core if $G[C]$ is a k-core and $|C| = h$. We call this an h-k-core.*

Definition 5 (Size-Constrained k-Core Group Query (SCCGQ)). *Given an undirected weighted graph $G = (V, E, W)$, a size constraint h, a coreness k, and a query user node u_q, a size-constrained k-core group query (SCCGQ) q can be modeled as a triple (u_q, h, k), which aims to find a user group of h nodes (including the query user u_q) that has the highest social closeness while also being a k-core.*

Example 1. Take Fig.1 as an example. G_1 is an original social graph displayed in Fig.1(a). Assume that an SCCGQ q is denoted as $(v_5, 4, 2)$, which means v_5 is the query user, the group size constraint is 4, and the coreness is 2. There are two induced subgraphs shown in Fig.1(b) and Fig.1(c). S_1 is an induced subgraph of $\{v_3, v_4, v_5, v_7\}$ in G_1, and its closeness $Co(S_1)$ is the sum of the weights of all the edges in S_1, that is, $Co(S_1) = 1.99$. Analogously, S_2 is an induced subgraph of $\{v_2, v_3, v_4, v_5\}$ in G_1, and its closeness $Co(S_2)$ is 1.16. The two subgraphs are both 4-2-core structures, and both include v_5. Finally, S_1 is the result of this SCCGQ query case, as S_1 has the highest closeness among all the groups that constitute 4-2-core structures and include v_5.

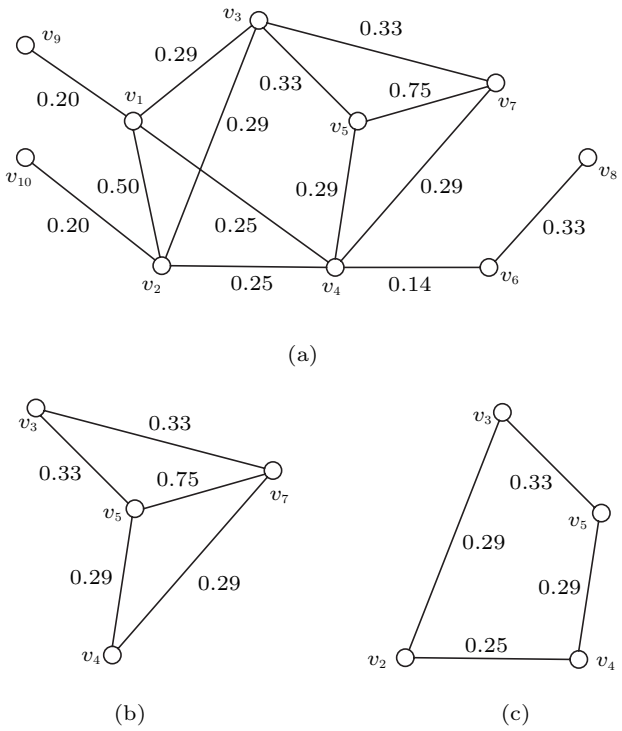


Fig.1. Example of the SCCGQ problem. (a) Social network G_1. (b) Result S_1. (c) Result S_2.

Theorem 1. *SCCGQ is NP-hard.*

Proof. We establish the hardness by a reduction from a well-known NP-hard problem, namely, the online Steiner tree (OST) problem.

The Steiner tree problem (ST) is a network design problem defined on a graph with edge costs. Its input is a set of terminals that need to be connected to one another. The online version of ST is the OST problem, in which the terminals are revealed at one time and each terminal needs to be connected to the current tree before the next one arrives. Given an instance of OST, we construct an instance of an SCCGQ $q = (u_q, h, k)$. We consider the restricted case of SCCGQ when $k = 1$. Note that SCCGQ aims to find a connected subgraph including u_q. Thus, the first terminal in the OST instance is u_q. The other terminals (users in SCCGQ) are revealed at one time. As the size constraint h is one part of the input of SCCGQ, the hardness cannot be reduced. The k-core structure (in SCCGQ) can be reduced to a tree structure (in OST) in polynomial time. Thus, we have OST$\preceq_P$SCCGQ. □

In this paper, we tackle the problem of efficiently processing SCCGQ in practical settings.

3 Query Processing

A naive solution to the SCCGQ problem can be readily described as follows. First, we leverage the k-core decomposition algorithm[17] to obtain a k-core of the original social graph G, denoted as G_k, in which the coreness of each node is no less than k. Second, we examine each subset of G_k with size h. Those subsets whose induced graphs constitute k-cores and include the query node can be seen as candidates. Finally, the naive solution selects the candidate with the highest closeness as the result of SCCGQ.

Given a social graph $G = (V, E, W)$, $|V| = n$, and $|E| = m$, in the worst case, the k-core of G is itself. Additionally, the total number of combinations needed to be enumerated is C_{h-1}^{n-1}, where $C_{h-1}^{n-1} = \frac{(n-1)!}{(h-1)!\times(n-h)!}$. The time complexity of the naive solution is $O(m+n^h \times h^2)$, where $O(m)$ is the time complexity of k-core decomposition, $O(n^h)$ is the time of selecting h nodes from the k-core G_k, and $O(h^2)$ is the time of estimating whether the induced subgraph of the set of the h nodes constitutes a k-core. When h is a constant, SCCGQ can be solved in polynomial time, while SCCGQ has exponential complexity when h is a part of the input[12].

To address the SCCGQ problem, we first propose an efficient algorithm, Blast Scatter (BS), in Subsection 3.1. We propose an effective social distance prun-

ing strategy in Subsection 3.2, which takes both the social graph topology and the query group size constraint into account. In Subsection 3.3, we deduce the upper bound of the closeness of a group. To speed up BS, we propose an advanced search algorithm, called Bounded Extension (BE), that combines the social distance pruning strategy and the upper bound.

3.1 Blast Scatter Algorithm

Apparently, the naive solution is extremely time consuming and inefficient. To overcome these deficiencies and achieve a better efficiency, we propose an efficient algorithm, Blast Scatter (BS), which appoints the query node as a center to start outward expansions. That is, BS does breadth search starting from the query node to perform expansions. In each expansion, BS selects multiple neighbors of the center node and selects a new node as the new center using a greedy strategy. When the center node is the query node, the number of selected neighbors lies in $[k, h]$, where k is the coreness and h is the size constraint. When the center node is not the query node, there are two cases. 1) If the number of neighbors of the center node (d_{center}) in the current expansion P is less than k, then the number of selected neighbors lies in $[k - d_{\text{center}}, h - |P|]$. 2) Otherwise, the number of selected neighbors lies in $[1, h - |P|]$. In both cases, we utilize a method based on breadth-first search (BFS)[18] to obtain all possible expansions.

When the size (i.e., the number of nodes) of an expansion reaches h, BS analyzes whether the induced subgraph of the h nodes in the original social network constitutes a k-core. BS then computes the social closeness. After all possible expansions have finished, the induced subgraph with the highest closeness is returned as the result of SCCGQ. BS leverages the graph topology to perform an expansion by selecting neighbors of the center node. Thus, the induced subgraph of each completed expansion must be a connected subgraph and include the query node.

Although we take the BFS algorithm as a departure point for our BS algorithm, our algorithm greatly improves upon and has many differences from the traditional BFS algorithm. 1) The BS algorithm selects multiple nodes at a time in each expansion. 2) The BS algorithm leverages a strategy to determine how many nodes are to be expanded in each step.

The first phase of BS adopts a k-core decomposition algorithm to obtain the k-core structure G_k of the original social graph, which is the same as the first phase of the naive solution. In the rest of this paper, discussions are restricted to G_k, unless otherwise stated. We focus uniquely on static social networks. In addition, the weight of the social edge will not change with the process of k-core decomposition.

Theorem 2. *Given a k-core G_k of the original social graph and an SCCGQ $q = (u_q, h, k)$, in each expansion, the search space of the candidate results is at most $C_k^d \times C_{h-(k+1)}^{n-d-1} + C_{k+1}^d \times C_{h-(k+2)}^{n-d-1} + \cdots + C_{h-1}^d$ if $d \geqslant h-1 \geqslant k$, while the search space is at most $C_k^d \times C_{h-(k+1)}^{n-d-1} + C_{k+1}^d \times C_{h-(k+2)}^{n-d-1} + \cdots + C_{h-(d+1)}^{n-d-1}$ if $h-1 > d \geqslant k$. Here, n is the size of G_k, i.e., the node number of G_k is n, d is the degree of the query user in G_k, and $C_k^d = \frac{d!}{k! \times (d-k)!}$.*

Proof. When $d \geqslant h-1 \geqslant k$, to guarantee that the degree of the nodes in the result is no less than k, at least k nodes in the neighbors of the query user should be selected. Meanwhile, at most $h-1$ nodes of the neighbors can be selected. If k nodes are selected from the d neighbors, we only need to select $h-(k+1)$ nodes from the other $n-d-1$ nodes. Therefore, there are $C_k^d \times C_{h-(k+1)}^{n-d-1}$ combinations. Because d is not less than $h-1$, the d neighbors can satisfy the size constraint of the SCCGQ problem without selecting more nodes from the non-neighbors of u_q in G_k. In this way, there are C_{h-1}^d possible combinations.

Analogously, the proof process can be derived when $h-1 > d \geqslant k$. □

Algorithm 1 shows the pseudo code of the BS algorithm. BS maintains two priority queues UQ and DQ. UQ stores the intermediate results that have not been extended. DQ maintains the intermediate results that have been extended. Additionally, the nodes will be put into DV if they cannot be included in any other expansions except those that have already lain in UQ and DQ. The BS algorithm performs expansions beginning from the query node, i.e., UQ is initialized as $(u_q, 0)$. In each expansion, BS selects the node with the highest degree as the new center (line 12). By Theorem 2, BS then selects multiple neighbors of the center at a time. In this step, BS should guarantee that the degree of the center node in the induced subgraph of this expansion is no less than k and the subgraph size does not exceed h (lines 14-17). Once the size of an expansion reaches h, this expansion has already been extended. The BS algorithm puts it into DQ and updates the current result (lines 18–21). The following example illustrates how the BS algorithm works.

Algorithm 1. Blast Scatter (BS) Algorithm

Input: weighted undirected graph $G = (V, E, W)$, positive integer k, size constraint h, query user u_q
Output: an h-k-core group with the highest closeness

```
1  Initialize result set S ← ∅;
2  Initialize the closeness of current result set Co ← 0;
3  Initialize priority queue UQ ← (u_q, 0), DQ ← ∅, DV ← ∅;
4  G_k ← G;
5  while {v ∈ G_k : k > |deg(v)|} ≠ ∅ do
6      G_k ← G_k/{v ∈ G_k : k > |deg(v)|};
7  if |G_k| < h or u_q ∉ G_k then
8      Algorithm termination;
9  while UQ ≠ ∅ do
10     H ← UQ.dequeue();
11     if |H| < h then
12         Select the node v ∈ H with the highest degree
           and v ∉ DV;
13         DV ← DV ∪ v;
14         for each set VP ⊆ N(v)/H, k ⩽ |VP| + d_H(v)
           and |VP| + |H| ⩽ h do
15             Induce a subgraph G_i of H ∪ VP in G_k;
16             if G_i ⊄ UQ ∪ DQ then
17                 UQ ← UQ ∪ ⟨G_i, Co(G_i)⟩;
18     if |H| = h and H ⊄ DQ then
19         DQ ← DQ ∪ H;
20         if {v ∈ H : k > |deg_H(v)|} = ∅ and Co(H) > Co
           then
21             Co ← Co(H), S ← H;
22 Return S;
```

Example 2. Take G_1 in Fig.1 as an example. Assume an SCCGQ $q = (v_5, 4, 2)$, the steps of the BS algorithm are shown in Fig.2. Initially, the result S is assigned as $\emptyset$, and its closeness Co is zero. The 2-core structure of G_1 is shown in Fig.2(a). The expansion will begin from the query user v_5. The degree of v_5 in the 2-core of G_1 is 3, as shown in Fig.2(b). Thus, there are four kinds of combinations, as shown in Fig.2(c). Once the node v_5 has already been extended, BS transforms the status of v_5. That is, BS puts v_5 into DV and changes the color of v_5 to be gray. Meanwhile, BS computes the closeness of the above four intermediate results and updates UQ.

In the next step, BS selects the intermediate result with the highest closeness in UQ to perform more expansions. As shown in Fig.2(c), c_1 is selected. It can be seen that the size of c_1 reaches the size constraint 4. Thus, BS updates the current result S as the node set of c_1 and its closeness Co. Then, BS puts c_1 into DQ, which indicates that c_1 is not needed to perform more expansions. In the following step, c_3 in Fig.2(c) is selected to perform more expansions, as it has the highest closeness in UQ. The expansion process of c_3 is shown in Fig.2(d). d_1 and d_2 are intermediate results extended from v_3 in c_3. The color of v_3 is changed to gray. Next, BS puts d_1 and d_2 into UQ. Note that d_3, extended from v_7 in c_3, will not be put into UQ, as d_3 has already existed in DQ. When all intermediate results have already been extended, BS returns S as the result of SCCGQ.

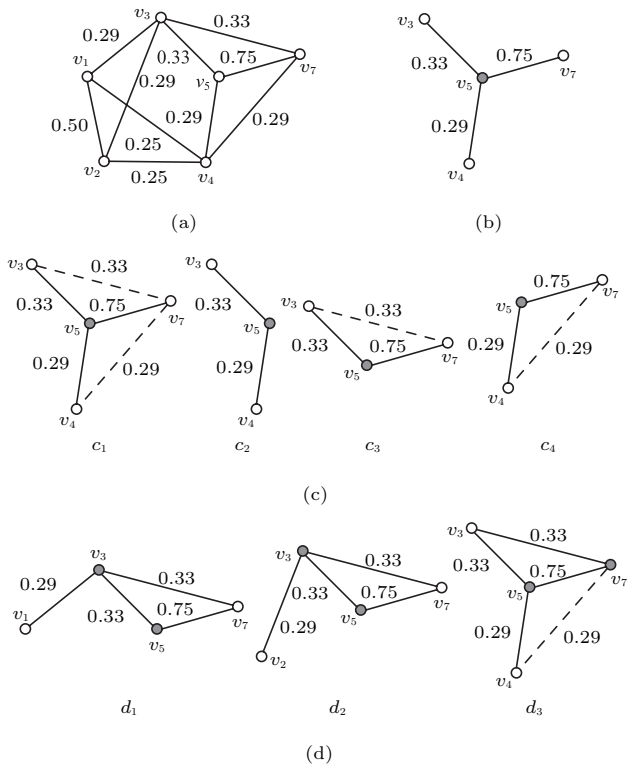


Fig.2. Steps of the BS algorithm. (a) k-core of G_1. (b) Query node. (c) Expansions from v_5. (d) Expansions from c_3.

3.2 Social Distance Pruning

The BS algorithm tackles the SCCGQ problem over the k-core of the original social network directly. The k-core is likely to be the original social network itself, as k is a part of the problem input. Thus, the reduction of the search space only by k-core decomposition may not be significant. If we can identify the nodes that must not be included in query results, then the search space and computational complexity could be greatly reduced. In this subsection, we design an effective pruning strategy based on social distance. In the rest of this paper, we regard the shortest social distance as the minimum edge number between two nodes, that is, we discuss social distance without considering the weight of edges.

Theorem 3. *Given $G = (V, E, W)$ and an SCCGQ $q = (u_q, h, k)$, $u_q \in G$, any $v \in V$ must not emerge in the query result if the shortest social distance from v to u_q is no less than $(h-1)$.*

Proof. Suppose v is one of the nodes in the final result S. The shortest social distance from v to u_q in the induced subgraph of S is no less than $(h-1)$. Note that any social path with length $(h-1)$ contains h nodes. Thus, the size of S has already reached h. If we want to guarantee that the degree of v in the induced subgraph of S is at least k, $k \geqslant 2$, we need to add more neighbors of v into S. The size of S will violate the query group size constraint. Therefore, the hypothesis is untenable, that is, v must not emerge in the query result.

Based on Theorem 3, an effective social distance pruning strategy can be proposed to accelerate SCCGQ queries. Given an SCCGQ $q = (u_q, h, k)$, we first construct a BFS tree rooted at the query user u_q. Those nodes with a tree height of at least $(h-1)$ can be pruned directly (the tree height starts from 0 at the root). With node deletions using this pruning strategy, the degree of the other nodes in the original graph will be changed. After that, the k-core decomposition can be used to prune more nodes.

Example 3. Take the social graph shown in Fig.3(a) as an example. Assume an SCCGQ $q = (v_1, 4, 3)$, a BFS tree rooted at v_1 is constructed as shown in Fig.3(b). By Theorem 3, nodes with tree height no less than $(4-1)$ can be pruned, that is, v_7 can be pruned directly. Moreover, v_8 can be pruned by the k-core decomposition, as the degree of v_8 in Fig.3(a) is changed from 3 to 2 after deleting v_7. As shown in Fig.3(c), the k-core is constructed after the social distance pruning.

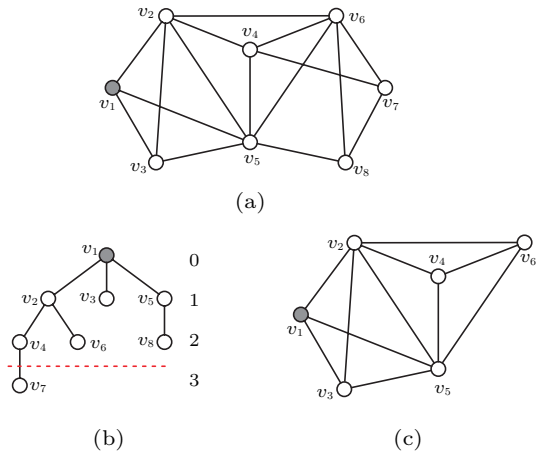


Fig.3. Steps of example 3. (a) Social graph. (b) BFS tree rooted at v_1. (c) k-core structure.

3.3 Bounded Extension

As shown in Algorithm 1, the BS algorithm puts the intermediate results of each expansion into a priority queue. If some expansions can be previously identified that they cannot be included in the final query result, the search space will be reduced. Consequently, the query processing can be accelerated. In this subsection, we first deduce the upper bound of closeness with respect to a user group. Then, we develop an advanced search algorithm, called Bounded Extension (BE), that combines the social distance pruning strategy and the upper bound to reduce the search space considerably.

Lemma 1. *Given a social graph $G = (V, E)$, let P be a user group in G and the size of P be $|P|$ ($|P| < h$). We denote the edge of $G[P]$ (remember that $G[P]$ is the induced subgraph of P in G) as $E(P)$. If $G[P]$ can be included in an h-k-core, then the number of additional nodes is $(h - |P|)$, and the number of additional edges is at most $C_2^{h-|P|} + (h - |P|) \times |P|$.*

Proof. By following Definition 4, the size of an h-k-core is h. Thus, the number of additional nodes is $(h - |P|)$ if the induced subgraph of P can be included in an h-k-core. Except for the edges in $G[P]$, the additional edges should consist of two parts. The first part contains the connections between any two nodes in a set with size $(h-|P|)$. The number of edges in this part is at most $C_2^{h-|P|}$, i.e., any two nodes in the $h-|P|$ set have a connection. Meanwhile, the second part contains the edges that connect the nodes in a set with size $h - |P|$ and the nodes in P. Analogously, the number of edges in the second part is at most $(h-|P|) \times |P|$. □

Theorem 4. *Given a social graph $G = (V, E)$, let P be a user group in G. The size of P is $|P|$ ($|P| < h$). We denote the edge of $G[P]$ (remember that $G[P]$ is the induced subgraph of P in G) by $E(P)$, and $Co(G[P])$ is the closeness of $G[P]$. Suppose that the maximum weight in G is $\omega_{\max}$, then the upper bound of the closeness with respect to this h-k-core is $(C_2^{h-|P|} + (h - |P|) \times |P|) \times w_{\max} + Co(G[P])$.*

Proof. From Lemma 1, if a user group P with a size less than h can be included in an h-k-core, then the edge number of this h-k-core is at most $C_2^{h-|P|}+(h-|P|)\times|P|+|P|$. Since the maximum weight in the whole social graph is $w_{\max}$, the upper bound of the closeness with respect to this induced subgraph is $(C_2^{h-|P|} + (h - |P|) \times |P|) \times w_{\max} + Co(G[P])$. □

Based on Theorem 4 and the social distance pruning strategy, we propose an advanced search algorithm, namely, Bounded Extension (BE). Before the k-core

decomposition process, BE adopts the social distance strategy to reduce the search space. Then, for each intermediate result G_i in each expansion, BE compares $UC(G_i)$, the upper bound of the closeness of G_i, with the closeness of the current result Co. G_i can be pruned if $UC(G_i) < Co$. The pseudo code is showed in Algorithm 2.

Algorithm 2. Bounded Extension (BE) Algorithm

Input: weighted undirected graph $G = (V, E, W)$, positive integer k, size constraint h, query user u_q
Output: an h-k-core group with the highest closeness

```
1  Initialize result set S ← ∅, the closeness of current result set Co ← 0;
2  Initialize priority queue UQ ← (u_q, 0), DQ ← ∅, DV ← ∅;
3  Construct a BFS-tree rooted at u_q;
4  G_k ← G/{v ∈ G : tree height of v is no less than (h − 1)};
5  while {v ∈ G_k : k > |deg_{G_k}(v)|} ≠ ∅ do
6      G_k ← G_k/{v ∈ G_k : k > |deg_{G_k}(v)|};
7  if |G_k| < h or u_q ∉ G_k then
8      Algorithm termination;
9  while UQ ≠ ∅ do
10     H ← UQ.dequeue();
11     if |H| < h then
12         Select the node v ∈ H with the highest degree and v ∉ DV;
13         DV ← DV ∪ v;
14         for each set VP ⊆ N(v)/H, k ⩽ |VP| + d_H(v) and |VP| + |H| ⩽ h do
15             Induce a subgraph G_i with H ∪ VP in G_k;
16             if G_i ⊄ UQ ∪ DQ then
17                 Compute closeness upper bound UC(G_i);
18                 if UC(G_i) > Co then
19                     UQ ← UQ ∪ ⟨G_i, Co(G_i)⟩;
20                 else DQ ← DQ ∪ G_i;
21     if |H| = h and H ⊄ DQ then
22         DQ ← DQ ∪ H;
23         if {v ∈ H : k > |deg_H(v)|} = ∅ and Co(H) > Co then
24             Co ← Co(H), S ← H;
25 Return S;
```

Example 4. Take Fig.4 as an example. Assume an SCCGQ $q = (v_6, 5, 2)$, where the query user is v_6, the size constraint equals 5, and the coreness is 2. Assume that Fig.4(a) is a 2-core structure of the original social graph after social distance pruning and k-core decomposition processing.

In the i-th extension of the BE algorithm, the current result is a set of nodes $\{v_6,\ v_7,\ v_8,\ v_9,\ v_{10}\}$, and the induced subgraph of it in the k-core structure is illustrated in Fig.4(b). The closeness Co is the sum of the weights of the edges in the induced subgraph, i.e., $Co = 2.55$. The induced subgraph of an intermediate result, $S_i = \{v_4, v_5, v_6, v_7\}$, is given in Fig.4(c). The closeness $Co(S_i)$ equals 0.7. By Theorem 4, the closeness upper bound of S_i, denoted as $UC(S_i)$, can be estimated. For $w_{\max} = 0.45$, $UC(S_i) = (C_2^{h-|S_i|} + (h - |S_i|) \times |S_i|) \times w_{\max} + Co(S_i) = 2.50$, which is lower than the closeness of the current result. Thus, S_i can be pruned without further processing.

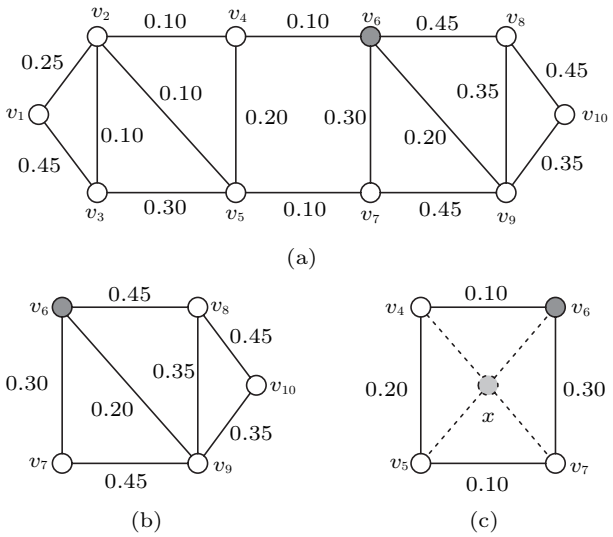


Fig.4. How the closeness upper bound works. (a) A k-core graph after SD-pruning. (b) Current result. (c) The i-th extension.

3.4 Query Optimization

To further accelerate the query processing, we develop an offline social-aware index (SAI).

Social-Aware Index (SAI). Given a coreness k, an SAI structure of a user u is a set in which any user group with u cannot be a k-core, denoted as $SAI_{u,k}$. We illustrate the idea of SAI in the following example.

Example 5. Consider the social graph in Fig.4. Given $k = 2$, the SAI of user v_6 is a set $\{v_1, v_2, v_3, v_4, v_5, v_9, v_{10}\}$. In this set, any user group with v_6 cannot be a 2-core. Note that for a given user and coreness, SAI may not be unique. For instance, the set $\{v_1, v_2, v_3, v_4, v_7, v_8, v_{10}\}$ is also an SAI of v_6 with $k = 2$.

Query Optimization Based on SAI. Given u and k, SAI indexes a user set in which any user group with u cannot be a k-core. Given an SCCGQ $q = (u_q, h, k)$, the expansions consisting of the query user and the nodes in SAI cannot be included in the final query result of q. We can delete these expansions directly without further processing. Therefore, the search space will be reduced considerably.

4 Experiments

In this section, we experimentally study the performance of the proposed algorithms. We perform a series of sensitivity tests to study the impact of query parameters using real social graph datasets. In the following, we first describe the experimental settings and then analyze the experimental results.

4.1 Experimental Settings

Algorithms. We study the performance of the following proposed algorithms and take the Naive algorithm elaborated in Section 3 as a baseline.

1) *BS.* The Blast Scatter (BS) algorithm is elaborated in Subsection 3.1. Based on the k-core of the original graph, BS appoints the query node as a center to begin outward expansions via a breadth search. In each expansion, BS selects a new center through a greedy strategy and then selects multiple neighbors of the center.

2) *BE.* The Bounded Extension (BE) algorithm is described in Subsection 3.3, which combines the social distance pruning strategy and the upper bound of closeness to reduce the search space.

Datasets. We evaluate the proposed algorithms on three real social graph datasets collected from Facebook, Brightkite, and Gowalla①. The properties of the three datasets are summarized in Table 1, where $|V_G|$ and $|E_G|$ represent the number of nodes and edges, respectively, $d_{\max}$ denotes the maximum degree, d_{ave} indicates the average degree, and *Dia* represents the diameter of the social graph, i.e., the distance of the longest shortest path. Note that the median degree is rather small due to the heavy tail of the power law degree distribution observed in these graphs. In addition, we construct synthetic social graphs using the R-MAT graph generator in the GT-Graph software[19], which has been widely used to generate power law social graphs[20].

Table 1. Some Statistics of Datasets

Dataset	$\lvert V_G\rvert$	$\lvert E_G\rvert$	$d_{\max}$	d_{ave}	*Dia*
Facebook	4 039	88 234	1 098	43.69	8
Brightkite	58 228	214 078	1 134	7.35	16
Gowalla	196 591	950 327	2 268	9.67	14

Setup. All experiments are implemented on a PC with CPU AMD Phenom™ II N830 Triple-Core Process (2.10 GHz), Memory 4.00 GB, SSD 128 GB. The operating system is Microsoft® Windows 7 Ultimate Edition. The development software is Microsoft® Visual Studio 2010, using the language C++ and its standard template library.

4.2 Experimental Results

We first test the performance of the proposed algorithms with the index optimization (elaborated in Subsection 3.4). The performance of the index will be analyzed in Subsection 4.2.5.

4.2.1 Efficiency of Different Algorithms

The objective of this set of experiments is to study the efficiency of the proposed algorithms with different real datasets. We set the value of the size constraint h to be 8, the coreness k to be 3, and the degree of the query user to be 10. Fig.5 shows the runtime of the three algorithms (Naive, BS, BE) on the three real datasets, respectively. It can be seen that the BE algorithm has the best efficiency among the three algorithms, and BS runs faster than the Naive algorithm. The reason is that the Naive algorithm adopts a brute force method, which is time consuming. The BS algorithm selects multiple neighbors of the center node at a time, and leverages the size constraint to retrain the number of selected neighbors in each expansion, which can accelerate the query processing. The reason why the BE algorithm runs faster than the BS algorithm is that the BE algorithm adopts the social distance pruning strategy and leverages the upper bound of closeness to reduce the search space.

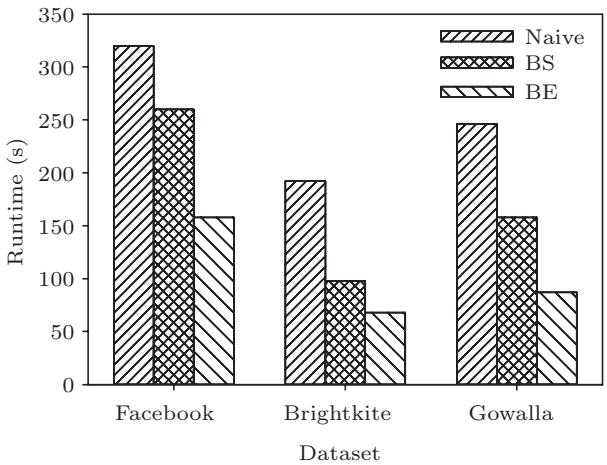


Fig.5. Efficiency.

①http://snap.stanford.edu/, Sept. 2018.

4.2.2 Evaluation of the Pruning Strategy

In this set of experiments, we take the BS algorithm and the BS+SD-pruning algorithm as contrast algorithms to study the efficiency of the social distance pruning strategy. The BS+SD-pruning algorithm is the BS algorithm with the social distance pruning strategy described in Subsection 3.2. The intuition of BS+SD-pruning is that it processes SCCGQ by adopting the social distance pruning strategy before k-core decomposition over the original social graph. We set the value for the size constraint h to be 8, the coreness k to be 3, and the degree of the query user to be 10. The result is shown in Fig.6. It can be seen that the performance of the BS algorithm is improved by more than 30% by leveraging the SD-pruning strategy.

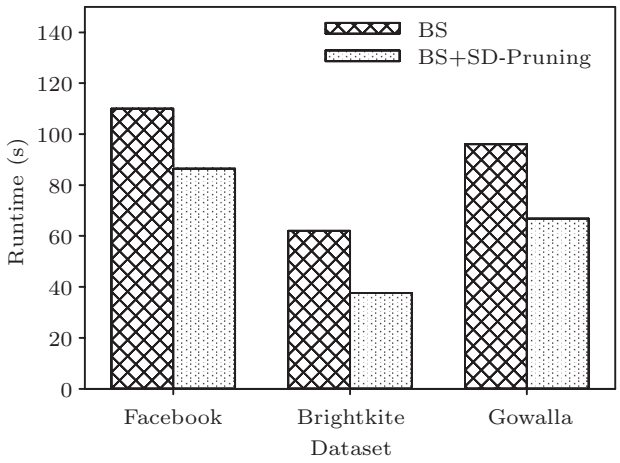


Fig.6. SD-pruning capability.

4.2.3 Scalability

We study the effect of graph size (vertex number) on the performances of our proposed algorithms using synthetic social graphs, which can be used to evaluate the scalability of our proposed algorithms.

We implement the Naive algorithm and the BE algorithm on synthetic datasets. Let the size constraint h be 10, the coreness k be 3, and the degree of the query user be 10. The results are shown in Fig.7. The runtime of the BE algorithm increases slowly with the number of vertices. Conversely, the runtime of the Naive algorithm increases rapidly. These all prove that the BE approach has good scalability.

4.2.4 Varying the Query Parameters

As mentioned in Section 2, an SCCGQ q is modeled as a triple (u_q, h, k), where u_q is the query user, h is the size constraint, and k is the coreness. We study the effect of the three parameters on the performance of our proposed algorithms. In particular, we regulate u_q for fixed coreness and size constraint pairs to estimate the effect of the degree of the query user. Similarly, we evaluate the impact of coreness (or the size constraint) by fixing the other two parameters.

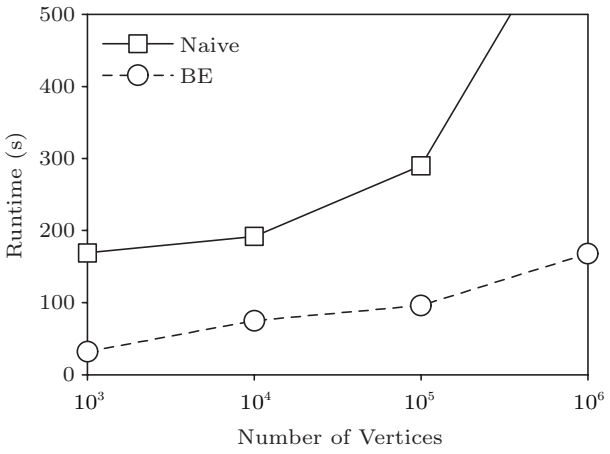


Fig.7. Testing scalability.

Varying the Size Constraint. In this set of experiments, we study the effect of the size constraint of a query. Let the coreness k be 3. We alter the size constraint h from 10 to 50. We regard the average runtime of 50 queries as the performance measure of our proposed algorithms, where the degree of query users is 9. Fig.8 shows the results on the Facebook, Brightkite and Gowalla datasets.

For fixed coreness and degree of the query user, the three algorithms run slower with the size constraint h increasing. The reason is that a query with a larger size constraint h is more likely to have a greater search space, that is, all the algorithms are required to process more intermediate results. It can also be seen that the runtime of the Naive algorithm increases rapidly as the size constraint h increases. The performance of the BS algorithm is preferable to that of the Naive algorithm. Finally, the runtime of the BE algorithm has a slow growth as the size constraint h increases, as it leverages the current best closeness and the upper bound of closeness to prune intermediate results.

Varying k. In this set of experiments, we study the effect of the coreness of a query. Let the size constraint h be 20, and we vary the coreness k from 2 to 10. We regard the average runtime of 50 queries as the performance measure of our proposed algorithms, where the degree of the query users is 10. The results on the three real datasets are shown in Fig.9. The three algorithms

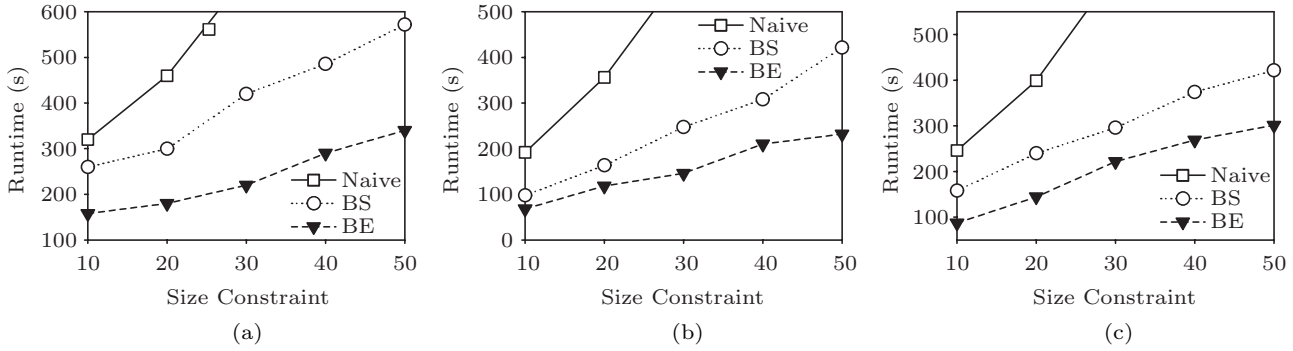


Fig.8. Varying size constraint h with $k = 3$. (a) Runtime test on the Facebook dataset. (b) Runtime test on the Brightkite dataset. (c) Runtime test on the Gowalla dataset.

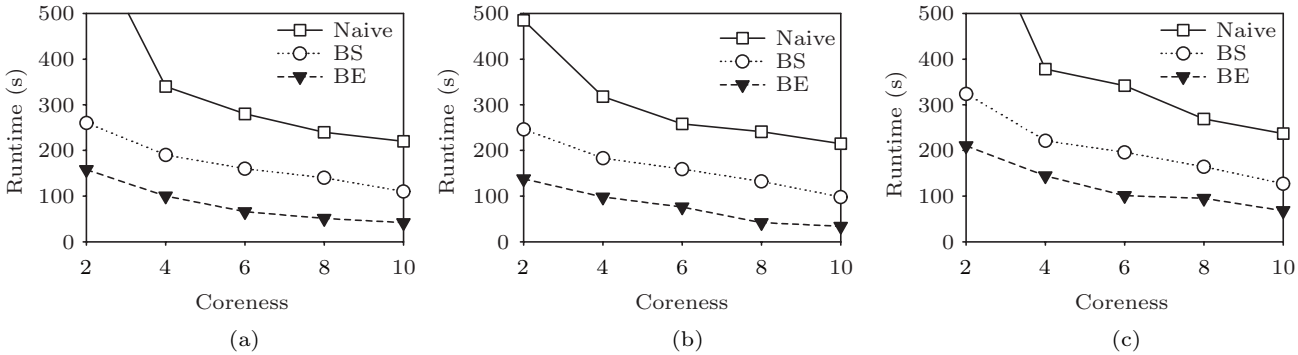


Fig.9. Varying coreness k with $h = 20$. (a) Runtime test on the Facebook dataset. (b) Runtime test on the Brightkite dataset. (c) Runtime test on the Gowalla dataset.

run faster as the coreness k increases. The reason is that a query with a larger coreness k is more likely to have a stronger social constraint, which indicates that the query has fewer intermediate results to process.

Varying the Degree of Query User. In this set of experiments, we study the effect of the degree with respect to the query user. Let the size constraint h be 20 and the coreness k be 6. We vary query users with different degrees from 10 to 50. Fig.10 shows the running time of the three algorithms over the three real datasets. For a fixed size constraint h and a fixed coreness k, the three algorithms run slower when the degree of the query user increases. The reason is that a larger query user degree leads to a larger search space. It also can be seen that the runtime of the Naive algorithm increases rapidly as the degree of the query user increases. Finally, the runtime of the BE algorithm grows slowly when the degree of the query user increases.

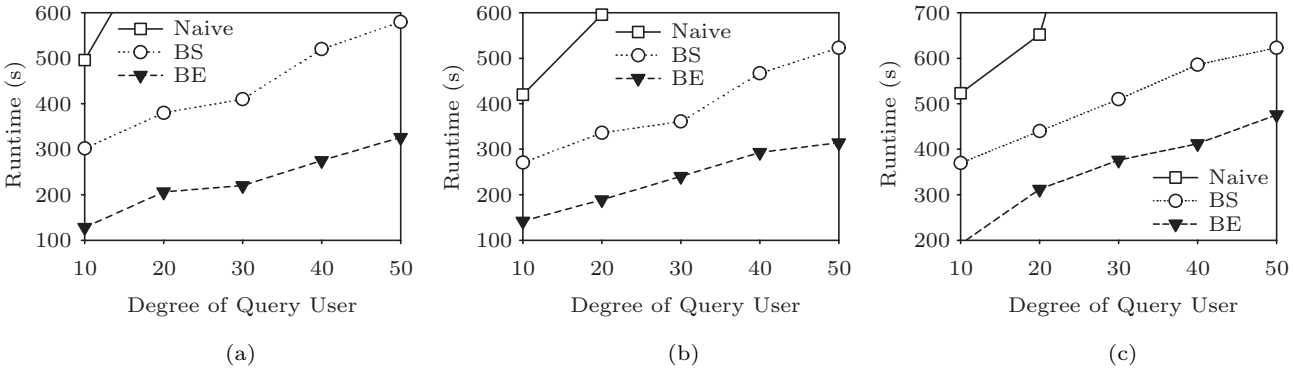


Fig.10. Varying the degree of query user with $h = 20$, and $k = 6$. (a) Runtime test on the Facebook dataset. (b) Runtime test on the Brightkite dataset. (c) Runtime test on the Gowalla dataset.

4.2.5 Index Performance

In this set of experiments, we study the cost of social-aware index (SAI) construction and the algorithmic efficiency based on SAI. Note that the construction of SAI is an offline procedure. It has relevance to the parameter k. As mentioned in Section 2, we address the problem of efficiently processing SCCGQ in practical settings. Thus, we construct SAIs for each user with $k \in [2, 10]$.

Table 2 shows the building time and storage cost of our index on the three datasets. To evaluate the pruning capability of our index, we implement the BE algorithm based on SAI, denoted as BE + SAI. We test the running time on the three datasets ($h = 20$, the degree of the query user is 10). The results are shown in Fig.11. It can be seen that the BE + SAI algorithm runs faster than the BE algorithm. The reason why SAI speeds up the query efficiency can be summarized as follows. $\text{SAI}_{u,k}$ maintains the users that cannot be part of the answer when $u_q = u$ and the coreness is k. Thus, the search space can be largely reduced by removing these users in $\text{SAI}_{u,k}$.

Table 2. Cost of SAI Construction

Dataset	Time (min)	Memory (MB)
Facebook	9.38	121.3
Brightkite	85.08	1 165.2
Gowalla	153.90	2 637.8

5 Related Work

With the rapid growth of social networks, close subgraph query[21] plays a key role in social network analysis and has attracted widespread attention. Many studies related to close subgraph query have become the focus of social network analysis. The main issue regarding close subgraph query is clique[22] or maximal clique[23]. Cheng *et al.*[24] presented the external-memory algorithm ExtMCE for maximal clique enumeration (MCE) computation on large real-world networks. ExtMCE recursively processes a small part of a large graph at a time and ensures that the set of max-cliques computed in the local steps is correct and complete in the whole graph. ExtMCE bounds the memory usage by the H^*-graph, a novel concept defined based on the notion of the h-index.

In a clique, any two nodes should have a social edge, which makes the formation condition very strict. Thus, many studies were derived from the relaxation of clique formation. Luce[25] presented a new concept named n-clique, a maximal subgraph of G, in which the distance in G between any two nodes is not larger than n. Seidman and Foster[26] proposed a concept of a k-plex, which consists of a set F of k nodes from a given graph G, and all the nodes in G should be covered in the neighbors of all the nodes in F. In addition, there are many other clique-like structures studied widely, which are omitted here due to the limited space.

Our work relates to the traditional k-core problem[16,27], which is one of the fundamental tasks in social network analysis. The k-core is the largest subgraph in which vertices have at least k interconnections. Core decomposition is essential for structure analysis and visualization of networks[28–30]. Cheng *et al.*[31] devised a novel top-down approach for core decomposition, which begins from the smallest-size core, i.e., the $k_{\max}$-core, and recursively reduces the search space and disk I/O cost for each k-core computation. Wen *et al.*[32] proposed an I/O efficient core maintenance algorithm to handle edge insertion, and an im-

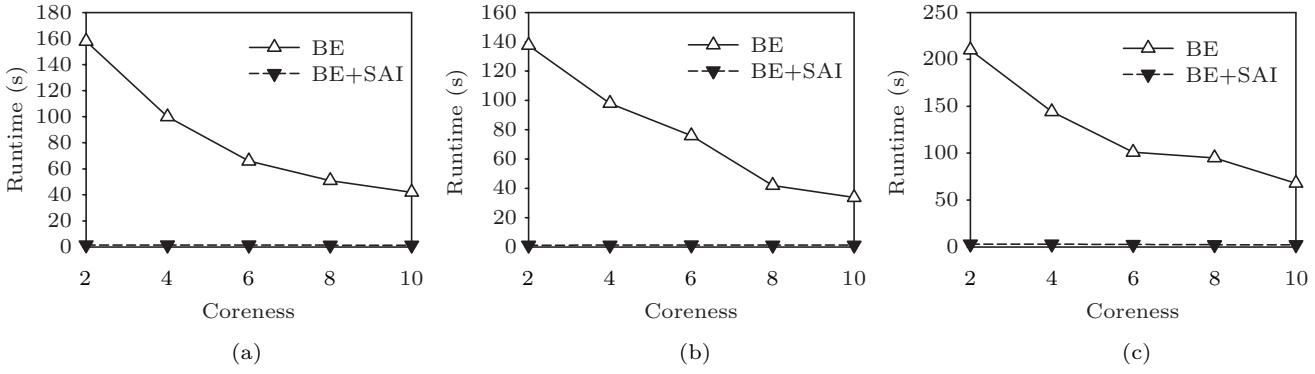


Fig.11. Performance of index. (a) Runtime test on the Facebook dataset. (b) Runtime test on the Brightkite dataset. (c) Runtime test on the Gowalla dataset.

proved algorithm to further reduce I/O and CPU cost by investigating some graph properties. From a double layer graph with social friendship and similarity, Zhang *et al.*[33] aimed to find all maximal (k, r)-cores where the similarity between each pair of selected nodes in the similarity layer is at least r. Existing work on k-core is different from our work, in which the query group is size-constrained by a parameter h. The parameter h and the coreness k are both part of the input, which makes our problem have exponential complexity. In addition, a k-truss of a graph G is the largest subgraph in which every edge is contained in at least $(k-2)$ triangles within the subgraph[34], which is also a specific structure for social network analysis and has already attracted much attention[35−37].

Recently, several studies have been developed for group queries. Deng *et al.*[38] studied the group nearest group query that finds a subset $\omega(|\omega| \leqslant k)$ of points from a given data point set D such that the total distance from all points in query point set Q to the nearest point in ω is not greater than the distance to any other subset $\omega'(|\omega'| \leqslant k)$ of points in D. This work focuses on the spatial distance between data points and query points, which is different from our work to find a user group with a specific social topology structure. In addition, Li *et al.*[39] proposed a novel type of geo-social query, which aims to find a minimum user group in which the members satisfy certain social relationships and the associated regions can jointly cover all the query points. Yang *et al.*[40,41] proposed two innovative social-spatial group queries. They focused on finding a set F of p vertices from a given graph G, and considered acquaintance and distance constraints. If the query is sent by diverse users with the same parameters, the results of these studies are identical to ours. Our work focuses on personalized user group queries, i.e., the results of different queries required by different users are variant. Our work is more appropriate for users who want to organize private activities, which makes full consideration of the unique social topology of query users.

6 Conclusions

In this paper, we studied the novel problem of a size-constrained k-core problem in a social network. We formally defined the problem as a size-constrained k-core group query (SCCGQ), which aims to find a user group of h nodes (including the query user) that has the highest social closeness while also being a k-core. To address the SCCGQ problem, we proposed a novel algorithm, Blast Scatter (BS), which appoints the query user as a center to begin outward expansions via a breadth search. In each outward expansion, BS selects a new center through a greedy strategy and then selects multiple neighbors of the center. Moreover, we proposed an advanced search algorithm, called Bounded Extension (BE), that combines an effective social distance pruning strategy and a tight upper bound of social closeness to prune the search space considerably. Experimental studies on large real-world datasets demonstrated the performance of our proposed algorithms.

Groups are a key characteristic of social networks, and the size-constrained group queries have significant influence in social network analysis and real-life applications. There are many potential future directions of this work. The social trust mechanism can be further explored to study group behaviors. In addition, it is interesting to investigate how to execute personalized group queries involving user preferences and group preferences.

References

[1] Allen J. Event Planning: The Ultimate Guide to Successful Meetings, Corporate Events, Fundraising Galas, Conferences, Conventions, Incentives and Other Special Events (2nd edition). Wiley, 2008.

[2] She J, Tong Y, Chen L, Cao C C. Conflict-aware event-participant arrangement and its variant for online setting. *IEEE Transactions on Knowledge and Data Engineering*, 2016, 28(9): 2281-2295.

[3] She J, Tong Y, Chen L, Song T. Feedback-aware social event-participant arrangement. In *Proc. ACM SIGMOD International Conference on Management of Data*, May 2017, pp.851-865.

[4] Tong Y, She J, Meng R. Bottleneck-aware arrangement over event-based social networks: The max-min approach. *World Wide Web*, 2016, 19(6): 1151-1177.

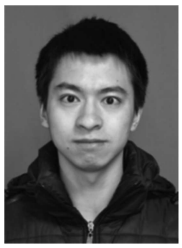
[5] Sinnen O, Sousa L A. Communication contention in task scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 2005, 16(6): 503-515.

[6] Tong Y, Wang L, Zhou Z, Ding B, Chen L, Ye J, Xu K. Flexible online task assignment in real-time spatial data. *Proceedings of the VLDB Endowment* 2017, 10(11): 1334-1345.

[7] Tong Y, She J, Ding B, Wang L, Chen L. Online mobile micro-task allocation in spatial crowdsourcing. In *Proc. the 32nd International Conference on Data Engineering*, May 2016, pp.49-60.

[8] She J, Tong Y, Chen L. Utility-aware social event-participant planning. In *Proc. ACM SIGMOD International Conference on Management of Data*, May 2015, pp.1629-1643.

[9] Tong Y, Chen L, Zhou Z, Jagadish H V, Shou L, Lv W. SLADE: A smart large-scale task decomposer in crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 2018, 30(8): 1588-1601.
[10] Scott J. Social Network Analysis (3rd edition). Sage, 2012.
[11] Knoke D, Yang S. Social Network Analysis (2nd edition). Sage Publishers, 2007
[12] Downey R G, Fellows M R. Fixed parameter tractability and completeness II: On completeness for $W[1]$. *Theoretical Computer Science*, 1995, 141(1/2): 109-131.
[13] Tan P N, Steinbach M, Kumar V. Introduction to Data Mining (1st edition). Pearson India, 2006.
[14] Guha S, Rastogi R, Shim K. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 2001, 25(5): 345-366.
[15] Spitzer F. Principles of Random Walk (2nd edition). Springer, 2001
[16] Seidman S B. Network structure and minimum degree. *Social Networks*, 1983, 5(3): 269-287.
[17] Khaouid W, Barsky M, Srinivasan V, Thomo A. K-core decomposition of large networks on a single PC. *Proceedings of the VLDB Endowment*, 2015, 9(1): 13-23.
[18] West D B. Introduction to Graph Theory (2nd edition). Pearson, 2000.
[19] Chakrabarti D, Zhan Y, Faloutsos C. RMAT: A recursive model for graph mining. In *Proc. the 4th SIAM International Conference on Data Mining*, April 2004, pp.442-446.
[20] Yuan Y, Lian X, Chen L, Yu J X, Wang G, Sun Y. Keyword search over distributed graphs with compressed signature. *IEEE Transactions on Knowledge and Data Engineering*, 2017, 29(6): 1212-1225.
[21] Hanneman R A, Riddle M. Introduction to Social Network Methods. University of California, 2005.
[22] Luce R D, Perry A D. A method of matrix analysis of group structure. *Psychometrika*, 1949, 14(2): 95-116.
[23] Bron C. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 1973, 16(9): 575-576.
[24] Cheng J, Ke Y, Fu A W C, Yu J X, Zhu L. Finding maximal cliques in massive networks by h*-graph. In *Proc. ACM SIGMOD International Conference on Management of Data*, June 2010, pp.447-458.
[25] Luce R D. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 1950, 15(2): 169-190.
[26] Seidman S B, Foster B L. A graph-theoretic generalization of the clique concept*. *Journal of Mathematical Sociology*, 1978, 6(1): 139-154.
[27] Dorogovtsev S, Goltsev A V, Mendes J F. K-core organization of complex networks. *Physical Review Letters*, 2006, 96(4): 040601.
[28] Alvarez-Hamelin J I, Dall'Asta L, Barrat A, Vespignani A. k-core decomposition: A tool for the visualization of large scale networks. arXiv:0504107, 2005. https://arxiv.org/abs/cs/0504107, May, 2018.
[29] Zhang H, Zhao H, Cai W, Liu J, Zhou W. Using the k-core decomposition to analyze the static structure of large-scale software systems. *The Journal of Supercomputing*, 2010, 53(2): 352-369.
[30] Saríyüce A E, Gedik B, Jacques-Silva G, Wu K L, Catalyürek Ü V. Streaming algorithms for k-core decomposition. *Proceedings of the VLDB Endowment*, 2013, 6(6): 433-444.
[31] Cheng J, Ke Y, Chu S, Özsu M T. Efficient core decomposition in massive networks. In *Proc. the 27th International Conference on Data Engineering*, April 2011, pp.51-62.
[32] Wen D, Qin L, Zhang Y, Lin X. I/O efficient core graph decomposition at web scale. In *Proc. the 32nd International Conference on Data Engineering*, May 2016, pp.133-144.
[33] Zhang F, Zhang Y, Qin L, Zhang W, Lin X. When engagement meets similarity: Efficient (k, r)-core computation on social networks. *Proceedings of the VLDB Endowment*, 2017, 10(10): 998-1009.
[34] Cohen J. Graph twiddling in a MapReduce world. *Computing in Science & Engineering*, 2009, 11(4): 29-41.
[35] Wang J, Cheng J. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment*, 2012, 5(9): 812-823.
[36] Huang X, Cheng H, Qin L, Tian W, Yu J X. Querying k-truss community in large and dynamic graphs. In *Proc. ACM SIGMOD International Conference on Management of Data*, June 2014, pp.1311-1322.
[37] Chen P L, Chou C K, Chen M S. Distributed algorithms for k-truss decomposition. In *Proc. the 2014 IEEE International Conference on Big Data*, October 2015, pp.471-480.
[38] Deng K, Sadiq S, Zhou X, Xu H, Fung G P C, Lu Y. On group nearest group query processing. *IEEE Transactions on Knowledge and Data Engineering*, 2012, 24(2): 295-308.
[39] Li Y, Chen R, Xu J, Huang Q, Hu H, Choi B. Geo-social k-cover group queries for collaborative spatial computing. *IEEE Transactions on Knowledge and Data Engineering*, 2015, 27(10): 2729-2742.
[40] Yang D N, Chen Y L, Lee W C, Chen M S. On social-temporal group query with acquaintance constraint. *Proceedings of the VLDB Endowment*, 2011, 4(6): 397-408.
[41] Yang D N, Shen C Y, Lee W C, Chen M S. On socio-spatial group query for location-based social networks. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2012, pp.949-957.



Yu-Liang Ma received his B.S. degree in computer science from the College of Computer Science and Engineering, Northeastern University, Shenyang, in 2013. Currently, he is a Ph.D. candidate of Northeastern University, Shenyang. His main research interests include graph databases, location-based social networks, and social network analysis.

Ye Yuan received his B.S., M.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, in 2004, 2007 and 2011, respectively. He is now a professor in the College of Computer Science and Engineering, Northeastern University, Shenyang. His research interests include graph databases, probabilistic databases, data privacy-preserving, and cloud computing.

Fei-Da Zhu is an associate professor at the School of Information Systems of Singapore Management University, Singapore. He obtained his Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, in 2009, and his B.S. degree in computer science from Fudan University, Shanghai, in 2001. His current research interests include data management and analytics, algorithms and complexity, very large scale pattern mining, and information network analysis.

Guo-Ren Wang received his B.S., M.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, in 1988, 1991, and 1996, respectively. Currently, he is a professor in the School of Computer Science and Technology, Beijing Institute of Technology, Beijing. His research interests are XML data management, query processing and optimization, bioinformatics, high-dimensional indexing, parallel database systems, and P2P data management.

Jing Xiao is the chief scientist and general manager of AI Engine Division of Ping An Technology (Shenzhen) Co., Ltd, Shenzhen. He received his Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, in 2005. His current research interests include artificial intelligence, big data, and analysis on finance and healthcare.

Jian-Zong Wang is a senior director of Artificial Intelligence at Ping An Technology (Shenzhen) Co., Ltd, Shenzhen. He obtained his Postdoctoral Fellow in artificial intelligence from the University of Florida, Gainesville, Florida, in 2015. He obtained his Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, in 2012. His current research interests include cloud computing, big data analysis, artificial intelligence, and deep learning.