Received September 5, 2020, accepted October 15, 2020, date of publication October 22, 2020, date of current version November 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032987

SGEQ: A New Social Group Enlarging Query With Size Constraints

XIAOXU SONG[1,3], BIN WANG[1], XIAOCHUN YANG[1], (Member, IEEE), JING QIN[1], LIANG ZHAO[2], AND LIANQIANG NIU[3]

[1]School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China
[2]College of Computer Science, Shenyang Aerospace University, Shenyang 110136, China
[3]College of Software, Shenyang University of Technology, Shenyang 110870, China

Corresponding author: Bin Wang (binwang@mail.neu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1700404, in part by the National Natural Science Foundation of China under Grant 61991404, in part by the Fundamental Research Funds for the Central Universities under Grant N171602003, in part by the Ten Thousand Talent Program under Grant ZX20200035, and in part by the Liaoning Distinguished Professor under Grant XLYC1902057.

ABSTRACT In recent years, the problem of k-core has attracted wide-spread research attention due to the popularity of the graph-related applications, such as social network analysis, community detection, and collaboration networks. To efficiently support group-based activity planning, the organizers need to guarantee the size and closeness of the group. Current applications of the k-core only support searching for the maximum k-core group or adding the minimum number of edges to obtain the k-core. However, no research has focused on the problem of enlarging the k-core. In reality, when new tasks arrive, a work team needs to not only recruit new team members based on the requirements but also guarantee sufficient closeness among the members for good cooperation. In this paper, we first formalize a new query, a social group enlarging query with size constraints (SGEQ), which aims to find n users to enlarge a subgraph from a k-core of size m to a $(k+\Delta)$-core of size $(m+n)$ by inserting the minimum number of edges. We prove that the SGEQ problem is NP-hard. To solve this problem, we first propose a novel algorithm, namely, the maximum connection edges algorithm (MCEA), which searches for the inserted vertex that has the most edges with an induced subgraph every time. Then, we develop an optimizing algorithm, namely, the maximum contribution degree algorithm (MCDA), which on average adds a number of edges in the expanded query less than that obtained by the MCEA. Finally, we conduct extensive experiments on two real-world datasets, and the results demonstrate the efficiency of the proposed algorithms.

INDEX TERMS k-core, social network, SGEQ, MCEA, MCDA.

I. INTRODUCTION

Most social network research has devoted serious attention to the construction of measures of the network structure. Many such measures have been defined, addressing a wide variety of network structural features [1]–[4]. The k-core plays an important role in identifying cohesive subgraphs, so it has been widely used in many graph-related applications, such as social network analysis, community detection, and collaboration networks. The k-core is defined as the maximal connected subgraph in which each vertex in the subgraph has at least k degrees. In recent years, many k-core studies have focused on k-core decomposition [5]–[14], k-core maintenance [15]–[18], [26], [27], maximizing the spread of influence [19]–[22], [31], [32], [35], and so on.

To guarantee to that the spread of influence is maximized, many studies have focused on maximizing the k-core [23]–[25]. To improve network stability, edge addition has been studied under different topics. [21] aims to add b edges to a subgraph to make it a k-core that is largest (with maximal vertexes) over the query graph. Indeed, [22] discusses adding the minimum number of edges to obtain a k-core subgraph containing at least n vertexes. However, these prior works have mostly studied how to find the maximum k-core subgraph by adding edges, whereas we focus on how to

The associate editor coordinating the review of this manuscript and approving it for publication was Chunsheng Zhu.

enlarge the existing k-core subgraph. The traditional k-core approaches cannot be directly applied to enlarge the k-core subgraph because there does not exist an effective strategy to add a minimum number of edges to enlarge the k-core subgraph. While their contributions are significant to the field, there is a gap with real-life applications that not only require the size of the group to be enlarged but also require the new group to establish a closer relationship that enlarges the core value, such as through collaborative team organization or event organization. Two motivating examples are listed below.

- **Collaborative team organization**. Suppose a research team has 5 people and that each member is familiar with at least 3 other members, making a 3-core in the group. The workload of the project has increased; accordingly, the team leader recruits 5 new members to accomplish the project by the deadline. To guarantee good cooperation, the team leader anticipates that the new team will make up a 5-core. However, the member recruited by the leader may not know 5 other members in the group. In this scenario, the k-core subgraph is enlarged to meet the need of the team leader for good cooperation.
- **Event organization**. Assume that an organizer wishes to hold a Lupus in Tabula game, that he expects 8 people to participate in this game, and that every participant is supposed to be familiar with at least 4 other participants. Since 4 participants on this team are familiar with each other, making up a 4-core, another 4 participants are needed to join the game. To attain better social benefits in this game, it is important that each participant should be familiar with at least 4 other participants. If this is difficult to achieve, minimum connections are added to meet the requirement.

This paper investigates the novel problem of a social group enlarging query with size constraints (SGEQ), which is based on enlarging an existing k-core subgraph. Intuitively, given an initial subgraph and a social network, an SGEQ query finds n users to enlarge the subgraph from a k-core with size m to a $(k+\Delta)$-core of size $(m+n)$ by inserting the minimum number of edges. Figure 1 illustrates an example SGEQ. The graph G contains 7 vertexes and their edges. The user supplies an initial subgraph G_k, in which the group of 3 vertexes v_1, v_2, v_3 constitute a 2-core. The SGEQ aims to find the user group G_{k+1}, which is a 3-core with a size of 5. Two optimal vertexes are selected to join the initial subgraph. Since the new subgraph is not a 3-core, an edge between u_1 and u_2 is inserted. By inserting the least number of edges, a 3-core subgraph is formed.

In this paper, we first formalize the novel problem of the social group enlarging query with size constraints (SGEQ). Formally, consider a social network undirected graph $G = (V, E)$, where $G_k(V_k, E_k)$ is the initial subgroup, which is a k-core subgraph of size $|V_k| = m$. The SGEQ extends $G_k(V_k, E_k)$ to a new $(k+\Delta)$-core subgraph $G_{k+\Delta}$ by adding new vertexes and new edges at the same time; meanwhile, $G_{k+\Delta}$ should satisfy the size constraint $|V_{k+\Delta}| = (m+n)$, and the added number of edges should be minimal.

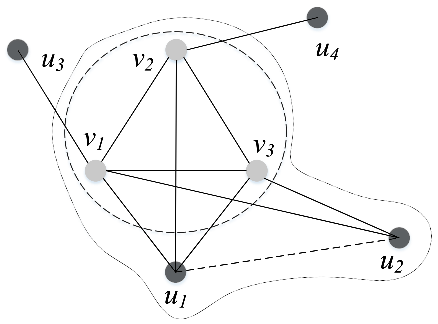


FIGURE 1. An example of an SGEQ.

To address the SGEQ problem, we first propose a baseline algorithm, namely, the maximum connection edges algorithm (MCEA), which searches for an optimal solution every time; i.e., the inserted vertex has the most edges with the induced subgraph. When a sufficient number of vertexes are inserted, all of the vertexes are sorted in ascending order of degree and the edges between two vertexes with the smallest degrees are iteratively inserted to update their degrees. This algorithm does not traverse all the vertexes in the entire graph and only finds the candidate set; that is, the vertexes have connected edges with the induced subgraph. However, this basic solution is still inefficient on graphs. These exists an issue that the selected optimal insertion vertex is always added to the subgraph, and this vertex may have a low effective contribution to the subgraph; that is, most of the vertexes connecting to the subgraph have degrees equal to or greater than $(k+\Delta)$. To address this shortcoming, we propose a novel algorithm, the maximum contribution degree algorithm (MCDA), which selects the inserted vertex in a way that efficiently increases the core value of the subgraph. The novelty of the MCDA lies in the idea of retrieving the candidate vertex that has the maximum contribution degree and ultimately adds the fewest edges to the subgraph.

The main challenge of simultaneously enlarging the size value and core value of an initial group is that the query retrieves highly qualified candidates to add the fewest edges to the subgraph to accomplish more tasks. Therefore, it is still difficult to design effective and efficient algorithms to find the optimal (or suboptimal) solutions. The major contributions of this work are summarized as follows:

- We formalize the social group enlarging query with size constraints (SEGQ). To the best of our knowledge, this is the first effort to define and address the SGEQ, which is proven to be NP-hard and intractable in practice.
- We design a research framework in which three stages are used to add the minimum number of edges to find the optimal team.
- We propose a baseline algorithm, namely, the maximum connection edge algorithm (MCEA), for efficient query processing of SGEQs.

- We further propose an advanced algorithm, namely, the maximum contribution degrees algorithm (MCDA), by retrieving highly qualified candidates. In contrast to the MCEA, the MCDA selects the candidate vertex in a way that efficiently increases the core value of the subgraph.
- We conduct extensive experiments to demonstrate the effectiveness and efficiency of our proposed algorithms.

The rest of this paper is organized as follows: We provide preliminaries and problem formulation of the social group enlarging query with size constraints in Section II. In Section III, a baseline maximum connection edge algorithm to solve SGEQ is described, and an improved algorithm is proposed, namely, the maximum contribution degree algorithm. The analysis of the experimental results is provided in Section IV. We discuss related work in Section V. We conclude the paper in Section VI.

II. PRELIMINARIES

In this section, we first introduce some background and definitions for our research. Then, we formalize the problem definition. Finally, we propose an overview of the research framework.

A. BACKGROUND AND DEFINITIONS

Consider a social network graph $G = (V, E)$, where the set of vertexes V denotes users and the set of edges E denotes the acquaintance relations among the users in V; for any two acquainted users, there exits $(u, v) \in E$. For a vertex $v \in V$, $deg_G(v)$ denotes the degree of v.

Density is an effective measure of network cohesion in social networks. To find the most cohesive group of acquaintances, we introduce the concept of the k-core as a social constraint to restrict the resulting group. In this section, we first present the definition and properties of the k-core, then propose the SGEQ problem.

Given an undirected social network graph $G = (V, E)$, a subgraph $G_k = (V_k, E_k)$ of G, where G_k is a k-core and $|V_k| = m$, and a vertex set $V_k = (v_1, \ldots, v_m)$, the cohesive core is an extended subgraph $G_{k+\Delta} = (V_{k+\Delta}, E_{k+\Delta})$, where $|V_{k+\Delta}| = (m + n)$. When vertex u in G is added to G_k, the contribution degree of u is defined as $degree^+(u) = \sigma(u, G_k)+\gamma(u, G_k)$. $\sigma(u, G_k)$ denotes the number of vertexes with edges connected to u in G_k whose degrees are less than $(k + \Delta)$. $\gamma(u, G_k)$ denotes the effective number of edges between u and G_k. E_c denotes the number of connected edges between u and the vertexes in G_k. The following cases exist: (1) if $\gamma(u, G_k) < (k + \Delta)$, $\gamma(u, G_k) = E_c$; (2) if $\gamma(u, G_k) \geqslant (k + \Delta)$, $\gamma(u, G_k) = (k + \Delta)$.

Definition 1: (K-core) [11]. Given a graph G and an integer k, the maximal connected subgraph $G' = (V', E')$, which is in G, is a k-core if $degG'(v) \geq k$ ($\forall v \in V'$).

Definition 2: (Coreness) [2]. 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.

TABLE 1. Summary of Notation.

Notation	Definition
$G(V, E)$	An undirected and unweight social network graph
u, v	A user in G
(u, v)	A user edge in G
$N(v)$	The set of neighbours of v in G
$deg(v)$	The degree number of v in G
$G_k(V_k, E_k)$	A social network subgraph that is a k-core graph such that $\|V_k\| = m$
$G_{k+\Delta}(V_{k+\Delta}, E_{k+\Delta})$	A social network extension subgraph that is a $(k+\Delta)$-core graph such that $\|V_{k+\Delta}\| = m+n$
$degree^+(v)$	The contribution degree of v
E_c	The number of connected edges between u and the vertexes in G_k
V_c	A candidate set of vertexes
$\gamma(u, G_k)$	The effective number of edges between u and G_k
$\sigma(u, G_k)$	The number of vertexes with edges connected to u in G_k whose degrees are less than $(k+\Delta)$
E_{min}	The minimum number of inserted edges
V_0	$V_0 = V - V_k$
V_r	n vertexes selected from V_0

Definition 3: (Minimum Added Edge Cost (MAEC)). Given an undirected social network graph $G = (V, E)$ and a subgraph $G_k = (V_k, E_k)$ of G, where G_k is a k-core and $|V_k| = m$, the cohesive core is an extended subgraph $G_{k+\Delta} = (V_{k+\Delta}, E_{k+\Delta})$ that satisfies the following conditions:

- $|V_{k+\Delta}| = (m + n)$;
- $\forall v \in V_{k+\Delta}, deg(v) \geq (k + \Delta)$;
- $argmin(|E_{k+\Delta}| - |E_k|)$.

Definition 4: (Maximum Contribution Degree Vertex (MCDV)). Given an undirected social network group $G = (V, E)$, a subgraph $G_k = (V_k, E_k)$ of G, where G_k is a k-core and $|V_k| = m$, and a vertex set $V_k = (v_1, \ldots, v_m)$, we intend to find an extension subgraph $G_{k+\Delta} = (V_{k+\Delta}, E_{k+\Delta})$, where $|V_{k+\Delta}| = m + n$. The MCDV is defined as $degree^+_{max}(u_i) = max\{\sigma(u_i, G_k) + \gamma(u_i, G_k)\}, u_i \in G$.

B. PROBLEM STATEMENT

In this section, we formally define the SGEQ problem.

Problem Statement. (Social Group Enlarging Query with Size Constraints(SGEQ)). Given an initial subgraph $G_k(V_k, E_k)$, which is a k-core subgraph of size $|V_k| = m$, the SGEQ finds a user result group of size $|V_{k+\Delta}| = (m + n)$ such that the new subgraph is a $(k + \Delta)$-core and the minimum number of edges is added. An SGEQ is represented as $Q = (G_{k+\Delta}(V_{k+\Delta}, E_{k+\Delta}), m + n)$ $(\Delta > 0)$, where $(k+\Delta)$ is the degree of every vertex in the extension subgraph $G_{k+\Delta}(V_{k+\Delta}, E_{k+\Delta})$ as a social aquaintance constraint and $|V_{k+\Delta}| = (m + n)$ is the size of the extension subgraph.

Theorem 1: *SGEQ* is NP-hard.

Proof : We prove that the SGEQ is an NP-hard problem by reduction from the c-clique, which is a classical NP-hard problem. Assume two problems:

P_1: Deciding whether there is a $(k - 1)$-clique in a graph G is *NP*-complete.

P_2: Given a subgraph g, we can decide whether we can add k vertexes in G and the incident edges, so that the new subgraph is an m-core ($m = k - 1$) group.

We can reduce the task in P_1 to that in P_2, given G, and construct the following graph G':

(1) add g, where g is an $(m-1)$-core graph;

(2) choose any node u in g and add an edge from u to every vertex in G.

Claim: P_2 is true if and only if P_1 is true. We first prove the necessity condition.

Proof: If P_2 is true, then $\exists V_1, V_2, \cdots, V_m \in G$ s.t. $g \cup \{V_1, V_2, \cdots, V_m\}$ is an m-core graph, and the edges incident on V_i can only be between $\{V_1, V_2, \cdots, V_m\}$ and u. Since each V_i's edge number is at least m, each V_i has $(k-1)$ edges to $\{V_1, V_2, \cdots, V_m\}$.

Let $k - 1 = m$, which means that the subgraph consisting of $V_1, V_2, \cdots, V_m$ is an m-clique.

$\Rightarrow P_1 = true$.

We then prove the sufficient condition. If P_1 is true, then $\exists V_1, V_2, \cdots, V_m \in G$ that form an m-clique. Therefore, if we add them to g (constructed the same way as above), each vertex in $g \cup \{V_1, V_2, \cdots, V_m\}$ has at least m edges within itself, such that

(1) for $V_1, V_2, \cdots, V_m$, each has $m + 1 = k$ edges to $g \cup \{V_1, V_2, \cdots, V_m\}$;

(2) for $u_i \in g$, u_i has $(m-1)+1 = m$ edges, since g is the $(m-1)$-core.

$\Rightarrow P_2 = true$.

The theorem is verified.

C. OVERVIEW OF THE RESEARCH FRAMEWORK

Fig.2 shows the research framework of the proposed algorithms for SGEQ, which primarily consists of three stages: extracting data, enlarging G_k, and determining the minimum number of inserted edges. First, we extract a k-core initial subgraph G_k of size m from the social network graph and find all vertexes that are connected to this subgraph as a candidate set V_c. When the subgraph changes, the candidate set is also updated. Second, we enlarge a subgraph from the k-core of size m to the $(k + \Delta)$-core of size $(m + n)$ by inserting the minimum number of edges. The process of expansion is to first enlarge the size of G_k by selecting the optimal vertex in the candidate set and inserting it into G_k each time until the size of the subgraph is $(m + n)$. The AMEA function enlarges the core value of G_k to $(k + \Delta)$. Finally, we obtain the minimum number of inserted edges.

III. THE HEURISTIC ALGORITHMS

In this section, we propose two algorithms based on the research framework in Figure 2. Since G_k is the initial subgraph, which is a k-core of size m, SGEQ aims to find a $(k + \Delta)$-core subgraph by adding new vertexes and new edges at the same time; meanwhile, $G_{k+\Delta}$ should satisfy the size constraint $|V_{k+\Delta}| = (m + n)$ and the added number of edges should be minimal. However, in practical problems, it is difficult to enlarge the size and core value of the subgraph at the same time. The algorithm is divided into two parts. Thus, an idea to efficiently find n users is to first begin solving the problem, which enlarges the size of the subgraph, and then insert the minimum number of edges to satisfy the $(k + \Delta)$-core. In the first part, we propose an algorithm, namely, maximum connection edges algorithm (MCEA), which searches for an optimal solution every time; i.e., the inserted vertexes have the most edges with the induced subgraph. When we insert a sufficient number of vertexes, all of the vertexes are sorted in ascending order according to degree, and we iteratively insert edges between two vertexes with the smallest degrees to update their degrees. Next, we propose an optimization algorithm, the maximum contribution degree algorithm (MCDA), which inserts vertexes that have the maximum contribution degrees and ultimately inserts fewer edges on average than the MCEA. The improvement to the MCDA is that every time the optimal insertion vertex selected is added to the subgraph, this vertex has a more effective contribution to the subgraph; that is, most of the vertexes connecting the subgraph have degrees less than $(k + \Delta)$. The second part is to insert the least number of edges. We propose the adding minimum edges algorithm (AMEA).

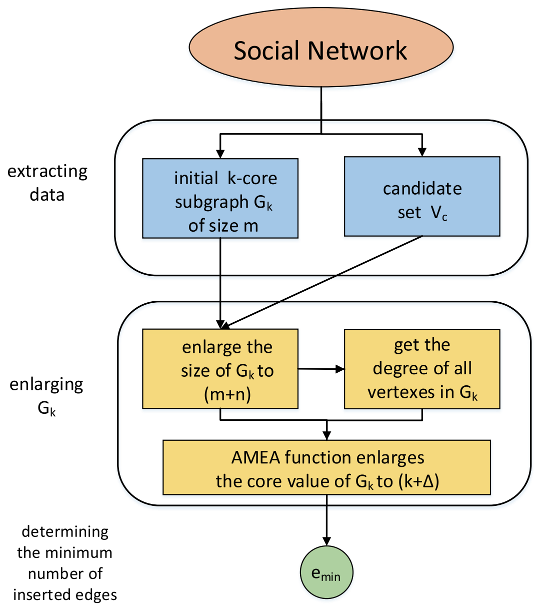

Social Network
extracting data
initial k-core subgraph G_k of size m
candidate set V_c
enlarging G_k
enlarge the size of G_k to (m+n)
get the degree of all vertexes in G_k
AMEA function enlarges the core value of G_k to (k+Δ)
determining the minimum number of inserted edges
e_{min}

FIGURE 2. An overview of the research framework.

A. MAXIMUM CONNECTION EDGES ALGORITHM

Since the operations are finding a subgraph and inserting edges on the graph, the latter algorithms are based on a k-core decomposition. We first regard G_k with size $|V_k| = m$ and the coreness of k as an induced subgraph. Second, we select some vertexes as candidate vertexes to set V_c, and the optimal vertex is chosen from V_c so that G_k in G forms a new subgraph $G_{k+\Delta}$. Finally, if the new subgraph $G_{k+\Delta}$ does not satisfy the coreness of $(k + \Delta)$, we add some edges that are not in G so that the new subgraph meets the requirements.

Given a social induced subgraph, the size of the subgraph is m and it has a k-core. The optimal S_0 vertexes are selected from G to G_k. The new subgraph $G_{k+\Delta}$ satisfies the following conditions: (1) $|V_{k+\Delta}| = (m+n)$; (2) $\forall v \in V_{k+\Delta}$, $deg(v) \geq (k+\Delta)$; (3) $argmin(|E_{k+\Delta}| - |E_k|)$.

The pseudocode of the MCEA algorithm is presented in Algorithm 1. According to the given induced subgraph, we select all the vertexes from G that have edges with the induced subgraph as the candidate set V_c. Each time we add a vertex with the greatest number of edges from the candidate set V_c to V_k (lines 4-9), we have to determine whether the number of vertexes in the subgraph was $(m+n)$ before adding the vertex (line 3). If the number of vertexes in the subgraph is $(m+n)$, we stop adding vertexes. If the number of vertexes in the subgraph is less than $(m+n)$, we iteratively add the vertex with the greatest number of edges. After adding the vertex, we update the subgraph G_k and the candidate set V_c (line 9). The above steps are repeated until the number of vertexes in the subgraph is $(m+n)$. Then, the details of the selection of candidate sets will be described, the vertexes will be selected in the subgraph, and each neighbour vertex will be compared with the vertexes in the subgraph. If a neighbour vertex is in the subgraph, we continue to access the next neighbour vertex. If the neighbour vertex is not in the subgraph, the neighbour vertex degree is increased by 1. After the neighbour vertexes of the vertexes in the subgraph are visited in turn, the neighbour vertex with the greatest degree is the optimal vertex in the subgraph. The vertex with the largest degree is added to the subgraph to form a new subgraph. Then, we select the neighbour vertex as a new candidate set V_c according to the new subgraph. After the size of the new subgraph reaches $(m+n)$, we add some edges to the subgraph satisfying the $(k+\Delta)$-core. Algorithm 2 describes the details of AMEA function. Finally, we return the new subgraph $G_{k+\Delta}$ and insert the minimum edges min_{add} (line 13).

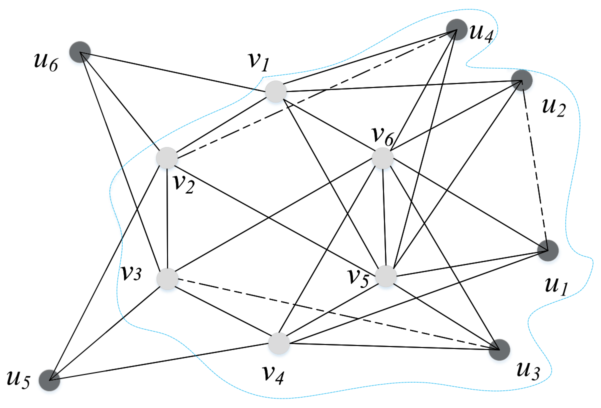


FIGURE 3. An example of the MCEA.

Example 1: The example of graph G, which contains twelve vertexes, is shown in Fig. 3. According to Definition 1, the user provides an initial subgraph G_k such that the group of six vertexes including $v_1, \ldots, v_6$ constitutes a 3-core. The MCEA returns the user group G_{k+1}. We select four vertexes

Algorithm 1 MCEA

```
Input: $G_k(V_k, E_k)$ is the initial subgraph which is a
       $k$-core subgraph with size $|V_k| = m$, the core
       $(k+\Delta)$ of enlarged subgraph, the size $(m+n)$ of
       enlarged subgraph;
Output: enlarged subgraph $G_{k+\Delta}(V_{k+\Delta}, E_{k+\Delta})$,
       the number of minimum inserted edges $min_{add}$;
1  $V_c \leftarrow$ all the neighbour vertexes of $G_k$;
2  $V_{k+\Delta} \leftarrow V_r + V_k$;
3  while $len(V_k) \leqslant (m+n)$ do
4      Find $v_j$ that satisfies:
5      $max\left(\sum_{1\leq i\leq p, 1\leq j\leq q} e(u_i, v_j)\right)$
6      $(u_1, u_2, \ldots, u_p) \in V_k, (v_1, v_2, \ldots, v_q) \in V_c$;
7      $V_k \leftarrow V_k \cup v_j$;
8      $V_c \leftarrow V_c - v_j$;
9      update $G_k, V_c$;
10 AMEA$(G_k(V_k, E_k), k+\Delta)$;
11 $G_{k+\Delta} \leftarrow G'_k$;
12 $min_{add} \leftarrow e_{add}$;
13 return $G_{k+\Delta}$ and $min_{add}$ values;
```

u_1, u_2, u_3, u_4 in the candidate set to join the original subgraph. Since the new subgraph is not a 4-core, we insert the least number of edges according to the AMEA to make the subgraph a 4-core. The dotted lines are the newly inserted edges.

We introduce the AMEA function in Algorithm 1.The new subgraph is enlarged by the MCEA from a k-core to a $(k+\Delta)$-core with the minimum edges inserted. This process aims to make minimal modifications to the new subgroup. The details of the adding the minimum edge algorithm are described in Algorithm 2. First, we split the graph into s sets according to the degree of every vertex in the new subgraph, and the sets are represented as $C_1, C_2, \ldots, C_s$ in ascending order. Second, we determine whether the degree of every vertex is greater than $(k+\Delta)$. If the degree of each vertex is greater than $(k+\Delta)$, the new subgraph is a $(k+\Delta)$-core; if the new subgraph is not a $(k+\Delta)$-core, we insert the fewest edges to make the new subgraph form a $(k+\Delta)$-core. According to the number of vertexes in C_i, we have the following two cases:

- If the number of vertexes in C_i is an odd number (Algorithm 3), the maximum number of inserted edges in C_i should be *slen* (line 1). We make a full permutation of the set C_i and operate on each permutation C one at a time (line 2). In the set C, we determine whether odd-numbered vertexes and even-numbered vertexes are connected by edges (line 5). If there is no connected edge, an edge is inserted; if there is a connected edge, no operation is performed. We iteratively process each set of odd and even vertexes (lines 3-9). If the number of inserted edges is exactly equal to the maximum number of edges that can be inserted in set C(line 10), we terminate the loop (line 13) and deal with the last vertex in set C, which

Algorithm 2 AMEA

```
Input: $G_k(V_k, E_k)$ is the initial subgraph which is a
       $k$-core subgraph with size $|V_k| = m$,
       the $(k + \Delta)$-core of enlarged subgraph;
Output: the new subgroup $G'_k(V'_k, E'_k)$, which is a
        $(k + \Delta)$-core subgraph of size $|V'_k| = (m + n)$,
        the number of minimum inserted edges $e_{add}$;
1  split into $s$ sets according to the degree of all vertexes
   $(C_1, C_2, \ldots, C_s)$ and
   $deg(C_1) < deg(C_2) < \ldots < deg(C_s)$;
2  $e_{add}$=false;
3  if $degree(\forall v \in V_k) \geqslant (k + \Delta)$ then
4      if $e_{add}! = 0$ then
5          $e_{add} \leftarrow 0$;
6          $G_k$ is a $(k + \Delta)$-core;
7  else
8      $e_{add} = 0$;
9      while $degree(C_i \in V_k$ in the order$) < (k + \Delta)$ do
10         $length$=len$(C_i)$;
11         if $(length\%2) = 1$ then
12             $edge$=OddVertex$(C_i, V_k, length)$;
13             $e_{add} = e_{add} + edge$;
14             update $G_k$;
15         else
16             $edge$=EvenVertex$(C_i, V_k, length)$;
17             $e_{add} = e_{add} + edge$;
18             update $G_k$;
19 $G'_k \leftarrow G_k$;
20 return $G'_k$ and $e_{add}$;
```

Algorithm 3 OddVertex

```
Input: the $i$-th set $C_i$, vertex set $V_k$, $length$ is the length
       of set $C_i$;
Output: the number of edges $insert_e$ inserted into the
        $i$-th set $C_i$;
1  $slen = (length - 1)/2$, $insert_e \leftarrow 0$, $d \leftarrow$ deg$(C_i)$;
2  for each $C \in permutation(C_i)$ do
3      $odd = 1$, $even = 0$, $insert = 0$;
4      while $odd < length$ and $even < length$ do
5          if $C[odd]$ not in $neighbor(C[even])$ then
6              $Insert(C[odd], C[even])$;
7              $odd+ = 2$, $even+ = 2$, $insert+ = 1$;
8          else
9              $odd+ = 2$, $even+ = 2$;
10     if $slen == insert$ then
11         $C^* \leftarrow C$;
12         $insert_e \leftarrow insert$;
13         break;
14     else
15         if $insert < slen$ and $insert > insert_e$ then
16             $insert_e \leftarrow insert$;
17             $C^* \leftarrow C$;
18 for each $v \in C^*$ and $deg(v)=d$ do
19     if $(\exists deg(u) > d$ in the order$)$ not in $neighbor(v)$ then
20         Insert$(v, u)$;
21         $insert_e+ = 1$;
22 return $insert_e$;
```

is not connected to another vertex. The remaining vertex is connected to a vertex whose degree is greater than those of the remaining vertexes (lines 18-21). If the number of inserted edges is less than the maximum number of edges that can be inserted into set C and the number of inserted edges is greater than the number of insertion edges of the completed permutation C (line 15), we save the number of inserted edges and this permutation C. Otherwise, we discard this permutation C. Finally, if the number of inserted edges in all permutations C is not equal to the maximum number of inserted edges *slen*, we take out the maximum number of inserted edges and the permutation C (lines 16-17). We deal with the remaining vertexes in set C, which are not connected to other vertexes. All remaining vertexes are connected to other vertexes whose degrees are greater than those of the remaining vertexes (lines 18-21). We calculate the number of all inserted edges (Algorithm 2, line 12) and update G_k (Algorithm 2, line 14) after the degree of all the vertexes in C_i is increased by the number of variable edges (Algorithm 2, line 13).

• If the number of vertexes in C_i is even (Algorithm 4), the maximum number of inserted edges in C_i should be *slen* (line 1). We make a full permutation of the set C_i and operate on each permutation C one at a time (line 2). In set C, we determine whether odd-numbered vertexes and even-numbered vertexes are connected by edges (line 5). If there is no connected edge, an edge is inserted; if there is the connected edge, no operation is performed. We iteratively process each set of odd and even vertexes (lines 3-9). If the number of inserted edges is exactly equal to the maximum number of edges that can be inserted into set C (line 10), we terminate the loop (line 13) and return the number of inserted edges. If the number of inserted edges is less than the maximum number of edges that can be inserted into set C and the number of inserted edges is greater than the inserted edges of the completed permutation C (line 15), we save the number of inserted edges and this permutation C. Otherwise, we discard this permutation C. Finally, if the number of inserted edges in all permutations C is not equal to the maximum number of inserted edges *slen*, we take out the maximum number of inserted edges and the permutation C (lines 16-17). The remaining vertexes in set C are not connected to other vertexes. All remaining vertexes are connected to other vertexes whose degrees are greater than those of the remaining vertexes (lines 18-21). We calculate the number

Algorithm 4 EvenVertex

```
Input: the $i$-th set $C_i$, vertex set $V_k$, $length$ is the length
       of set $C_i$;
Output: the number of edges $insert_e$ inserted into the
        $i$-th set $C_i$;
1  $slen = length/2$; $insert_e \leftarrow 0$; $d \leftarrow \deg(C_i)$
2  for each $C \in permutation(C_i)$ do
3      $odd=1$, $even=0$, $insert=0$;
4      while $odd < length$ and $even < length$ do
5          if $C[odd]$ not in $neighor(C[even])$ then
6              Insert $(C[odd], C[even])$;
7              $odd+=2$, $even+=2$, $insert+=1$;
8          else
9              $odd+=2$, $even+=2$;
10     if $slen == insert$ then
11         $C^* \leftarrow C$;
12         $insert_e \leftarrow insert$;
13         goto 22;
14     else
15         if $insert<slen$ and $insert>insert_e$ then
16             $insert_e \leftarrow$ insert;
17             $C^* \leftarrow C$;
18 for each $v \in C^*$ and $deg(v)=d$ do
19     if ($\exists deg(u) > d$ in the order) not in $neighor(v)$ then
20         Insert$(v, u)$;
21         $insert_e+=1$;
22 return $insert_e$;
```

of all inserted edges (Algorithm 2, line 15) and update G_k (Algorithm 2, line 17) after the degree of all the vertexes in C_i is increased by the number of variable edges (Algorithm 2, line 16).

We iteratively perform the above steps until the degrees of all the vertexes are greater than or equal to $(k + \Delta)$; then, the algorithm is terminated. Finally, we return G'_k and e_{add}.

Theorem 2: Given a k-core G_k of size m, we aim to find a $(k + \Delta)$-core $G_{k+\Delta}$ of size $(m + n)$. After adding n users, G_k is expanded to G'_k, which is not a $(k + \Delta)$-core. The total number of inserted edges is the smallest iff one edge is inserted between the two vertexes in G'_k with the lowest degree[1] to make G'_k be a $(k + \Delta)$-core.

Proof : The essence of subgraph G'_k becoming a $(k + \Delta)$-core is to increase the degree of the vertexes less than $(k+\Delta)$. If an edge is inserted into G'_k, the degrees of all vertexes can change by at most 1. Assume that vertexes x and y are the two vertexes with the smallest degrees in subgraph G'_k and that there is no connection between them. To increase the degrees of x and y by 1, x and y need to add at least one edge, respectively, assuming (x, u) and (y, w). If we remove edges (x, u) and (y, w), we insert edge (x, y). We consider three cases.

For the $u, w \notin G'_k$ case, the total number of edges is reduced by 1, but the total degree of subgraph G'_k is unchanged.

For the $u \notin G'_k$, $w \in G'_k$ case, to keep the total degree of subgraph G'_k unchanged, vertex w needs to add another edge. If vertex w is connected to a vertex in G'_k, the total degree will increase.

For the $u, w \in G'_k$ case, to keep the total degree of subgraph G'_k unchanged, vertex w or vertex u needs to add another edge.

This shows that modifying the initial algorithm and inserting connected edges x and y is either equivalent to the initial algorithm or removes one edge. The proposition holds.

[1]We arrange the $(m+n)$ vertexes in ascending order of degree every time. If the first two vertexes have connected edges, we check whether the first and subsequent vertexes have connected edges until we find a vertex that has no connected edge with the first vertex. We update all vertex degrees and repeat this operation until all vertex degrees are greater than or equal to $(k + \Delta)$.

Algorithm 5 MCDA

```
Input: $G_k(V_k, E_k)$ is the initial subgraph which is a
       $k$-core subgraph with size $|V_k| = m$,
       the $(k + \Delta)$-core of enlarged subgraph, the size
       $m + n$ of enlarged subgraph;
Output: the enlarged subgraph $G_{k+\Delta}(V_{k+\Delta}, E_{k+\Delta})$,
        the minimum number of inserted edges $min_{add}$;
1  $V_c \leftarrow$ all the neighbour vertexes of $G_k$;
2  calculate the degree of every vertex in $G_k$;
3  while $len(V_k) \leqslant (m + n)$ do
4      $max = 0$;
5      for each $v \in V_c$ do
6          $cnt = 0$;
7          for each $u \in V_k$ do
8              if $degree(u) < (k + \Delta) \cap$
               $degree(u) < (k + \Delta)$ then
9                  $cnt \leftarrow cnt + 2$;
                   $degree(v) = degree(v) + 1$;
10             else
11                 if $degree(u) \geqslant (k + \Delta) \cap$
                   $degree(u) < (k + \Delta)$ then
12                     $cnt \leftarrow cnt + 1$;
                       $degree(v) = degree(v) + 1$;
13     if $cnt > max$ then
14         $max \leftarrow cnt$;
15         $v_{max} \leftarrow v$;
16     $V_k \leftarrow V_k \cup v_{max}$;
17     $V_c \leftarrow V_c - v_{max}$;
18     update $G_k$, $V_c$;
19 AMEA$(G_k(V_k, E_k), k + \Delta)$;
20 $G_{k+\Delta} \leftarrow G'_k$;
21 $min_{add} \leftarrow e_{add}$;
22 return $G_{k+\Delta}$ and $min_{add}$;
```

B. MAXIMUM CONTRIBUTION DEGREE ALGORITHM

The second algorithm is the maximum contribution degree algorithm (MCDA). The pseudocode of the MCDA algorithm is presented in Algorithm 5. The vertex with the greatest contribution degree is added to subgraph G_k each time. In contrast to the MCEA, the vertex that is added to the subgraph G_k is more efficient, and the contribution degree to the subgraph is higher. According to the given induced subgraph, we select all the vertexes in G that are connected to the induced subgraph as the candidate set V_c (line 1). The candidate with the most effective degree, which is selected from the candidate set V_c, is added to the subgraph. Before adding the vertex to G_k, we decide whether the number of vertexes in the subgraph G_k has reached $(m+n)$ (line 3). If the vertex number of the subgraph is $(m + n)$, we stop adding vertexes to G_k; otherwise, the vertex with the maximum contribution degree is iteratively added, then the vertexes in G_k and the candidate set V_c are updated after adding the candidate vertex (lines 18-20). The details of selecting the maximum contribution degree vertex are as follows: First, the degree of every vertex in the subgraph G_k is calculated (line 2). Second, the vertex contributions for all vertexes in the candidate set V_c for G_k are computed sequentially. Finally, the maximum contribution vertex is added to G_k. The vertex contributions are divided into the following three cases (lines 8-15):

- If $deg(u) < (k + \Delta)$, $u \in G_k$, and $deg(v) < (k + \Delta)$, $v \in V_c$, $(u, v) \in G$, then $degree^+(v) = +2$;
- If $deg(u) \geqslant (k + \Delta)$, $u \in G_k$, and $deg(v) < (k + \Delta)$, $v \in V_c$, $(u, v) \in G$, then $degree^+(v) = +1$;
- If $deg(u) \geqslant (k + \Delta)$, $u \in G_k$, and $deg(v) \geqslant (k + \Delta)$, $v \in V_c$, $(u, v) \in G$, then $degree^+(v)$ is unchanged.

All the vertexes in V_c are obtained in turn, their contribution degrees are calculated(lines 5-6), and the vertex with the maximum contribution to G_k is added (lines 16-18). The above steps to add candidate vertexes to the subgraph are performed iteratively until the number of vertexes in the subgraph is $(m + n)$. Here, the AMEA function of Algorithm 2 is applied. Finally, a new subgraph $G_{k+\Delta}$ is obtained, and the minimum number of edges e_{add} is inserted (line 22).

Example 2: An example of graph G, which contains twelve vertexes, is shown in Fig. 4. According to Definition 1, the user provides an initial subgraph G_k such that the group of six vertexes $v_1, \ldots, v_6$ constitutes a 3-core. The MCDA returns the user group G_{k+1}. Four vertexes u_1, u_2, u_3, u_5 in the candidate set are joined to the initial subgraph. Since the new subgraph is not a 4-core, according to the AMEA, the least number of edges is inserted to make the subgraph a 4-core. The dotted lines are newly inserted edges. The vertexes selected by the MCEA and MCDA are different. The minimum number of edges inserted by the MCEA is 3, while the minimum number of edges inserted by the MCDA is 2.

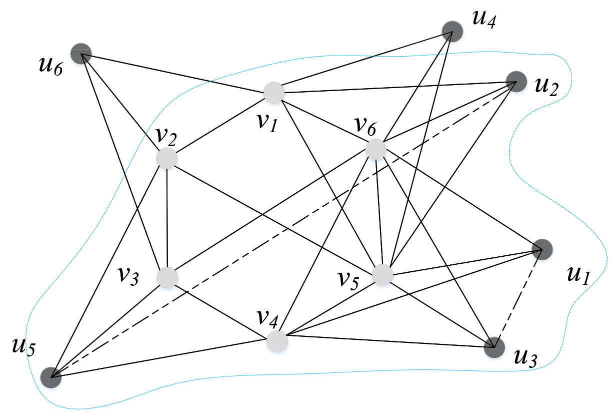


FIGURE 4. An example of the MCDA.

C. COMPLEXITY ANALYSIS

We first introduce a naive method to solve the SGEQ, and the details of the algorithm are as follows: We obtain a social network graph $G(V, E)$ such that $|V| = P$ and $|E| = Q$ and a k-core subgraph $G_k(V_k, E_k)$ such that $|V_k| = p$ and $|E_k| = q$. The new extension subgraph $G_{k+\Delta}$ $(V_{k+\Delta}, E_{k+\Delta})$ is $|V_{k+\Delta}| = i$, $|E_{k+\Delta}| = j$. We choose $(i - p)$ vertexes that are added from G_k to G'_k one at a time, so the total number of combinations that need to be enumerated is C^{i-p}_{P-p}, where $C^{i-p}_{P-p} = \frac{(P-p)!}{(i-p)!(P-i)!}$. The complexity of the naive algorithm is $O\left((i - p)!\right)$.

Since the complexity of the naive algorithm is too great to solve the SGEQ problem, it is impossible to calculate the result in polynomial time when the amount of graph data is extremely large. To solve the complexity problem, we propose two heuristic algorithms to solve the SGEQ in polynomial time. We adopt a novel algorithm, the maximum connection edges algorithm (MCEA), which searches for an optimal solution every time; i.e., the inserted vertexes have the most edges with the induced subgraph. When we insert a sufficient number of vertexes, all of the vertexes are sorted in ascending order according to degree and edges are iteratively inserted between two vertexes with the smallest degrees to update their degrees. The complexity of the MCEA is $O(p + q \times \overline{deg(V)} \times p)$, where p represents the size of the group, q represents the number of vertexes added to the group and $\overline{deg(V)}$ represents the average degree of the vertexes in the dataset. Another heuristic algorithm is the maximum contribution degree algorithm (MCDA), in which the selected insertion vertex has the maximum degree of contribution to the group each time. The complexity of the MCEA is $O(p + q \times \overline{deg(V_{contri})} \times p)$, where p represents the size of the group, q represents the number of vertexes added to the group and $\overline{deg(V_{contri})}$ represents the average contribution degree of vertexes in the dataset. In contrast to the MCEA, the MCDA selects the candidate vertex in a way that efficiently increases the core value of the subgraph. However, the MCDA runs slower than the MCEA since the MCDA calculates the maximum contribution degrees of the candidate vertexes.

IV. ANALYSIS OF THE EXPERIMENTAL RESULTS

In this section, we have conducted experimental studies using 2 real social graph datasets, and we study the performance of our algorithm by comparing it with two algorithms. We first introduce the experimental settings.

A. EXPERIMENTAL SETTINGS

Setup. All experiments are implemented on a machine with two Intel (R) Core (TM) i7-4790 (3.60 GHz) CPUs, 8.00 GB RAM, and 1 TB SSD. Our algorithms are implemented in Python and complied with JetBrains PyCharm Community Edition 2018.3.2. The operating system is Microsoft Windows 7 Ultimate Edition.

Algorithm. To the best of our knowledge, there are no related works that can solve the SGEQ problem in a social graph. In this paper, we propose two algorithms to slove the SGEQ problem, namely, the maximum connection edges algorithm (MCEA) and the maximum contribution degree algorithm (MCDA).

Datasets. We use two specific datasets, ego-Facebook and loc-Brightkite, to evaluate our proposed algorithms. To better obtain the initial subgraph, we made some changes to the ego-Facebook dataset. Both datasets used in the experiment are undirected graphs, which can be downloaded from SNAP (http://snap.stanford.edu/). The dataset properties are shown in Table 2, where $|V|$ represents the number of vertexes, $|E|$ represents the number of edges in the real social graph dataset, deg_{max} denotes the maximum degree, and deg_{ave} means the average degree. We selected the average vertex degree in the datasets to be 43.70 and 14.78. In this way, the difference in the number of edges of the initially selected graph is more helpful in the analysis of the experimental results.

TABLE 2. Real graph dataset properties.

Dataset	$\|V\|$	$\|E\|$	deg_{max}	deg_{ave}
ego-Facebook	4039	88234	1098	43.70
loc-Brightkite	58228	214078	1134	7.35

Parameters. The experiments are conducted using different settings of 4 parameters: k (the core value of the initial subgraph), m (the size of the initial subgraph), $k + \Delta$(the core value of the extension subgraph), and $m + n$ (the size of the extension subgraph). The parameter ranges are shown in Table 3. We set the range of k from 3 to 5. We set the range of m from 5 to 9. We vary $(k + \Delta)$ from 4 to 9. We vary $(m+n)$ from 10 to 16. According to the different initial graphs, we choose different $(m + n)$ and $(k + \Delta)$.

TABLE 3. Parameters of the Datasets.

Parameter	Range	Default
k	3, 4, 5	3
m	5, 7, 9	5
$k+\Delta$	4, 5, 6, 7, 8, 9	6
$m+n$	10, 11, 12, 13, 14, 15	11

B. EFFECTIVENESS

In this section, we verify the effectiveness of the two algorithms MCEA and MCDA for SGEQ problems in the datasets ego-Facebook and loc-Brightkite.

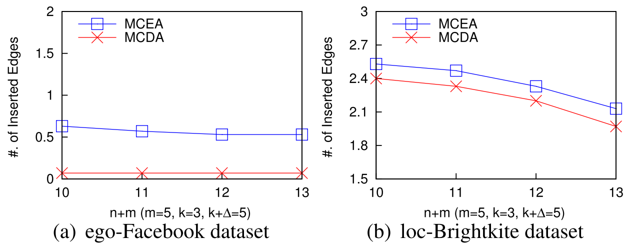


FIGURE 5. #. of Inserted Edges (MCEA vs. MCDA).

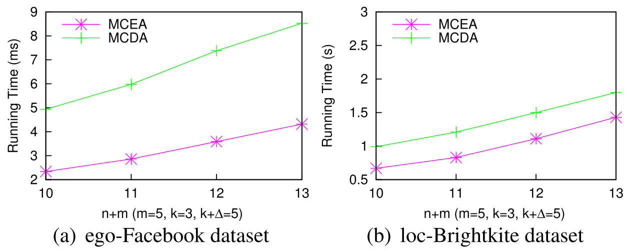


FIGURE 6. Running Time (MCEA vs. MCDA).

The initial subgraphs with size=5 and core=3 are formed into new subgraphs with different sizes, and the k values are enlarged. We set the newly expanded subgraphs, whose core values are 5 or 6, to range from 10 to 13 in Figures 5 and 7. After the number of inserted edges in ego-Facebook and loc-Brightkite are obtained, we can see that the average number of edges added is not large compared with the change in the size of the two algorithms, and they can effectively form the new subgraphs we need in Figure 5. With the increase in the core value of the new subgraph, we can see that the number of edges to be inserted has increased significantly in Figure 7, but the algorithms still work very well. We set the size of the new subgraph to 11 and 13 in Figures 9 and 11, respectively. As the value of k increases, the number of edges added increases accordingly.

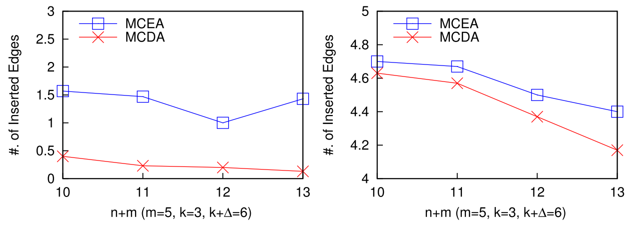
C. EFFICIENCY

In this section, we evaluate the efficiency of the two proposed algorithms. The number of inserted edges and the running time of the two algorithms are compared on the two datasets. We set the size of the initial subgraph m to be 5 and the value of the core to be 3. To solve the complexity problem, we propose two approximation algorithms to solve the SGEQ in polynomial time. We adopt a novel algorithm, the maximum connection edges algorithm (MCEA), which searches for an optimal solution every time; i.e., the inserted vertexes have the most edges with the induced subgraph. When we insert a sufficient number of vertexes, all of the vertexes are sorted in ascending order according to degree, and edges are iteratively inserted between two vertexes with the smallest degrees to update their degrees. Another algorithm is the maximum contribution degree algorithm (MCDA), in which the inserted vertexes have the maximum contribution degrees and fewer edges are ultimately inserted on average than with the MCEA algorithm.

1) EFFECT OF THE SIZE OF THE SUBGRAPH

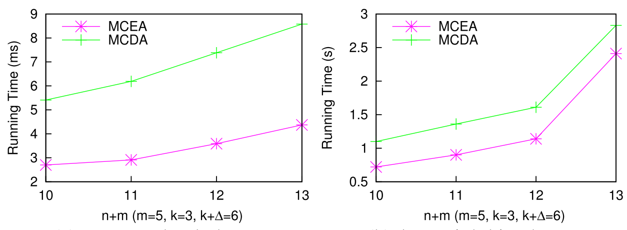
Figures 5-6 show the number of inserted edges and the running time for the two proposed algorithms (MCEA, MCDA) in the datasets ego-Facebook and loc-Brightkite when the enlarged core of the new subgraph is 5. We fix the core value to evaluate the effect of the graph size on the number of inserted edges in Figure 5 and the running time in Figure 6. We alter the size of the new subgraph n from 10 to 13. We use the average number of inserted edges and a running time of 30 queries to measure the performance of the proposed algorithm. In Figure 5, we find that the MCDA added fewer than the MCEA in both datasets, and this was more obvious in the dataset ego-Facebook. In this experiment, the average number of edges added in the dataset loc-Brightkite is more than that in the dataset ego-Facebook. The reason is that the average degree of vertexes on loc-Brightkite is smaller than that in ego-Facebook. In Figure 6, the two proposed algorithms run slower with increasing size. The reason is that when the size of the subgraph increases by 1, the number of times traversing the first hop of the subgraph will increase by 1. The MCDA is slower than the MCEA in running time because calculating the contribution degree of vertexes to the subgraph is slower than calculating the number of edges of the same vertexes to the subgraph.

We fix the core value to evaluate the effect of the graph size on the number of inserted edges in Figure 7 and the running time in Figure 8. When the core of the subgraph is 6, the trends of the two algorithms are similar to the results of the first set of experiments. The second set of data adds significantly more edges than the first set of data because the core value becomes larger. However, the increase in the running time of the two algorithms on the two datasets is not obvious.



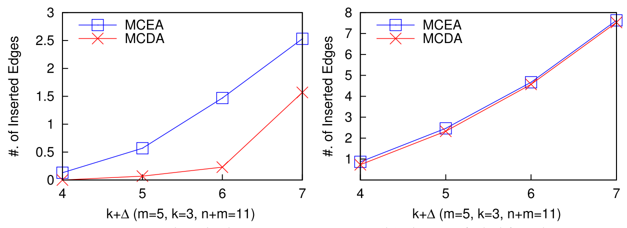
(a) ego-Facebook dataset (b) loc-Brightkite dataset

FIGURE 7. **#. of Inserted Edges (MCEA vs. MCDA).**



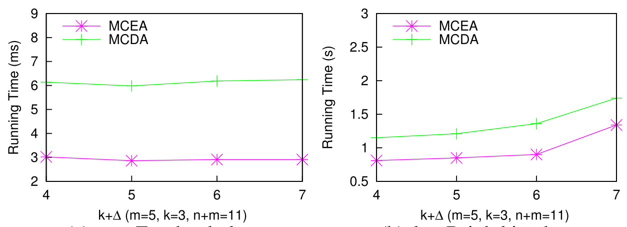
(a) ego-Facebook dataset (b) loc-Brightkite dataset

FIGURE 8. **Running Time (MCEA vs. MCDA).**



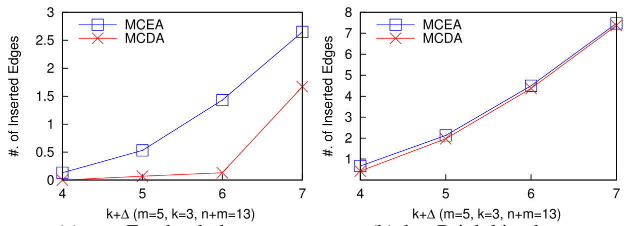
(a) ego-Facebook dataset (b) loc-Brightkite dataset

FIGURE 9. **#. of Inserted Edges (MCEA vs. MCDA).**



(a) ego-Facebook dataset (b) loc-Brightkite dataset

FIGURE 10. **Running Time (MCEA vs. MCDA).**



(a) ego-Facebook dataset (b) loc-Brightkite dataset

FIGURE 11. **#. of Inserted Edges (MCEA vs. MCDA).**

2) EFFECT OF THE NUMBER OF CORES

We fix the size value to evaluate the effect of the graph size on the number of inserted edges in Figures 9 and 11, and the running time in Figures 10 and 12. We alter the core value of the new subgraph from 4 to 7 when the enlarged size of the new subgraph is 11 and 13, respectively. We use the average number of inserted edges and a running time of 30 queries to measure the performance of the proposed algorithms. In the case of a fixed size, Figures 9 and 11 show that the number of edges inserted by the two algorithms will increase as the core value increases. Figures 10(a) and 12(a) show that the running time of the two algorithms will change little as the core value increases. Figures 10(b) and 12(b) show that the running time of the two algorithms will increase as the core value increases. In these four experiments, the average number of edges added in the dataset loc-Brightkite is greater than that in the dataset ego-Facebook. The reason is that the average degree of vertexes in loc-Brightkite is smaller than that in ego-Facebook.

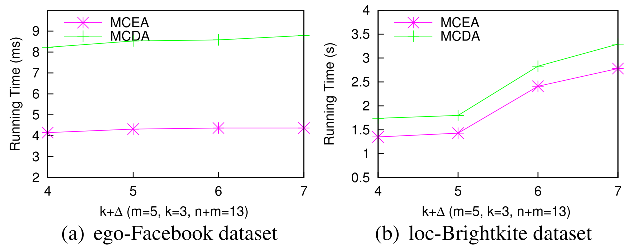


FIGURE 12. Running Time (MCEA vs. MCDA).

V. RELATED WORK

A. GRAPH PROCESSING

In graph theory, the core number of vertexes will be updated when edges are inserted or deleted. There are various existing works on core maintenance in a dynamic graph. Rong-Hua Li *et al.* [16] proposed a coloring algorithm by coloring and recoloring specific nodes to identify which core numbers need to be updated when edges of the graph are inserted or deleted. Unfortunately, it does not support partitioning graph data in parallel. Aksu *et al.* [15] proposed a new algorithm for k-core maintenance to update the original subgraph by incremental changes to the underlying graph and further improved the core maintenance algorithm in accommodating distributed k-core construction. Afterward, to effectively process the insertion or deletion of multiple edges, Jin *et al.* [18] proposed a parallel algorithm that can improve the efficiency of core maintenance in many real data sets. Moreover, faster parallel algorithms [26] have been devised to solve the core maintenance problems. The algorithms process all edges in the joint edge set in one iteration and greatly increase the parallelism and reduce the processing time. Recently, Liu and Zhang [27] improved the core decomposition algorithm to suit edge-weighted graphs. The algorithm finds a small subgraph that contains all vertexes after a change in a large graph. However, they do not focus on changing the topology of the graph. Some hypergraph modeling of social networks has also been studied. Fang *et al.* [31] proposed a novel topic-sensitive influencer mining framework that aims to find topical influential users and images. In a hypergraph, the hyperedges are utilized to capture multiple relations including, visual-textual content relations among images and social links between users and images. A time-varying hypergraph [32] has been developed to model an online social network using the time-varying features of multitype relationships, and an algorithm is developed to extract a domain-aware user trust network based on the time-varying hypergraph and user trust relationships. Kang *et al.* [35] proposed diffusion centrality, which is used to characterize vertexes that are influential in diffusing a property p. A heuristic algorithm, coarsened back and forth, is proposed to compute the top-k vertexes. The retrieval system returns meaningful result sets that are ranked by a deep neural network. However, this method focuses on maximizing the spread of influence. In contrast to the above three works, this mostly studies how to find the maximum k-core subgraph. For the edge k-core problem, [22] discusses adding the minimum number of edges to obtain a k-core subgraph containing at least n vertexes. However, it proposes graphs with bounded tree widths that cannot be extended to handle general graphs. Reference [21] proposes a heuristic algorithm with effective pruning techniques to make a k-core that is largest (with maximal vertexes) over the query graph. They enlarge the size of the k-core subgraph, but we focus on how to enlarge the existing k-core subgraph.

B. SOCIAL NETWORK ANALYSIS AND QUERY PROCESSING

There are many works on social search in a social network. The efficient processing of queries that consider both spatial and social relations is essential for LBSNs. Doytsher *et al.* [28] proposed a set of socio-spatial query operators to obtain valuable information from a socio-spatial graph. Li *et al.* [33] proposed the geo-social k-cover group query to minimize user groups. Each user is familiar with at least k other users in the group, and the user's coverage regions can jointly cover all query areas. For collaborative team organization, Sozio and Gionis [29] studied a query-dependent variant of the community search problem, which finds the query vertexes and incident edges with them in the graph. Yang *et al.* [30] proposed the social-temporal group query to find a group of attendees in which the sum of the geographical distance to the initiator is the minimum. The group of attendees also meets an acquaintance constraint so that the activity can run more smoothly. Zhang *et al.* [34] proposed extracting graph structure and node properties from social network subgraphs to prune candidate sets. He *et al.* [36] introduced a social retrieval mechanism to investigate novel deep neural networks for the ranking problem. However, they do not support enlarging k-core queries. Unlike the above social network queries, the SGEQ aims to find n users to enlarge a subgraph from a k-core of size m to a $(k + \Delta)$-core of size $(m + n)$ by inserting the minimum number of edges.

VI. CONCLUSION

The problem of enlarging k-core is widely used as graph-related applications such as collaborative team organization, event organization, and so on. In this paper, we formulate a novel query, namely, the social group extension query with size constraints (SEGQ), which aims to find n users to enlarge a subgraph from a k-core of size m to a $(k + \Delta)$-core of size $(m + n)$ by inserting the minimum number of edges. We carry out a systematic study of the SEGQ. First, we design a research framework in which three stages are used to add the fewest number of edges to find the optimal team. Second, we propose a baseline algorithm, MCEA, for the SEGQ problem. To overcome the shortcomings of the MCEA, we further propose a novel algorithm, MCDA, which selects the inserted vertex in a way that efficiently increases the core of the subgraph. Finally, we conduct extensive experiments on two real-world datasets, and the results demonstrate the efficiency and effectiveness of the proposed algorithms.

However, our research may lead to several new directions for future work. First, in many real scenarios, the new members added may be far from the meeting place and unable to arrive in time. Therefore, we should consider geographical factors in future research to facilitate the better organization of events. Second, in teamwork, we need to add members with specific identities or occupations to complete the work. Tag-based social networks will be studied in our next work.

REFERENCES

[1] S. B. Seidman, "Network structure and minimum degree," *Social Netw.*, vol. 5, no. 3, pp. 269–287, Sep. 1983.

[2] Y.-L. Ma, Y. Yuan, F.-D. Zhu, G.-R. Wang, J. Xiao, and J.-Z. Wang, "Who should be invited to my party: A size-constrained k-core problem in social networks," *J. Comput. Sci. Technol.*, vol. 34, no. 1, pp. 170–184, Jan. 2019.

[3] J. Wang and J. Cheng, "Truss decomposition in massive networks," 2012, *arXiv:1205.6693*. [Online]. Available: http://arxiv.org/abs/1205.6693

[4] L. Chang, "Efficient maximum clique computation over large sparse graphs," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 529–538.

[5] W. Khaouid, M. Barsky, V. Srinivasan, and A. Thomo, "K-core decomposition of large networks on a single PC," *Proc. VLDB Endowment*, vol. 9, no. 1, pp. 13–23, Sep. 2015.

[6] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 51–62.

[7] D. Wen, L. Qin, Y. Zhang, X. Lin, and J. X. Yu, "I/O efficient core graph decomposition at Web scale," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 133–144.

[8] A. Montresor, F. De Pellegrini, and D. Miorandi, "Distributed K-core decomposition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 2, pp. 288–300, Feb. 2013.

[9] A. Tripathy, F. Hohman, D. H. Chau, and O. Green, "Scalable k-core decomposition for static graphs using a dynamic graph data structure," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1134–1141.

[10] A. E. Saríyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and Ü. V. Çatalyürek, "Streaming algorithms for k-core decomposition," *Proc. VLDB Endowment*, vol. 6, no. 6, pp. 433–444, Apr. 2013.

[11] V. Batagelj and M. Zaversnik, "An O(m) algorithm for cores decomposition of networks," 2003, *arXiv:cs/0310049*. [Online]. Available: https://arxiv.org/abs/cs/0310049

[12] A. E. Sarıyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and Ü. V. Çatalyürek, "Incremental *k*-core decomposition: Algorithms and evaluation," *VLDB J.*, vol. 25, no. 3, pp. 425–447, Jun. 2016.

[13] D. Chu, F. Zhang, X. Lin, W. Zhang, Y. Zhang, Y. Xia, and C. Zhang, "Finding the best k in core decomposition: A time and space optimal solution," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 685–696.

[14] A. Caliò, A. Tagarelli, and F. Bonchi, "Cores matter? An analysis of graph decomposition effects on influence maximization problems," in *Proc. 12th ACM Conf. Web Sci.*, 2020, pp. 184–193.

[15] H. Aksu, M. Canim, Y.-C. Chang, I. Korpeoglu, and O. Ulusoy, "Distributed *k*-core view materialization and maintenance for large dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2439–2452, Oct. 2014.

[16] R.-H. Li, J. X. Yu, and R. Mao, "Efficient core maintenance in large dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2453–2465, Oct. 2014.

[17] Y. Zhang, J. X. Yu, Y. Zhang, and L. Qin, "A fast order-based approach for core maintenance," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 337–348.

[18] H. Jin, N. Wang, D. Yu, Q.-S. Hua, X. Shi, and X. Xie, "Core maintenance in dynamic graphs: A parallel approach based on matching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2416–2428, Nov. 2018.

[19] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2003, pp. 137–146.

[20] K. Bhawalkar, J. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma, "Preventing unraveling in social networks: The anchored *k*-core problem," *SIAM J. Discrete Math.*, vol. 29, no. 3, pp. 1452–1475, Jan. 2015.

[21] Z. Zhou, F. Zhang, X. Lin, W. Zhang, and C. Chen, "K-core maximization through edge additions," 2019, *arXiv:1906.12334*. [Online]. Available: http://arxiv.org/abs/1906.12334

[22] R. Chitnis and N. Talmon, "Can we create large *k*-cores by adding few edges?" in *Proc. Int. Comput. Sci. Symp. Russia*, 2018, pp. 78–89.

[23] R. Chitnis, F. V. Fomin, and P. A. Golovach, "Preventing unraveling in social networks gets harder," 2013, *arXiv:1304.6420*. [Online]. Available: http://arxiv.org/abs/1304.6420

[24] F. Zhang, W. Zhang, Y. Zhang, L. Qin, and X. Lin, "OLAK: An efficient algorithm to prevent unraveling in social networks," *Proc. VLDB Endowment*, vol. 10, no. 6, pp. 649–660, Feb. 2017.

[25] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Efficiently reinforcing social networks over user engagement and tie strength," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 557–568.

[26] Q.-S. Hua, Y. Shi, D. Yu, H. Jin, J. Yu, Z. Cai, X. Cheng, and H. Chen, "Faster parallel core maintenance algorithms in dynamic graphs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1287–1300, Jun. 2020.

[27] B. Liu and F. Zhang, "Incremental algorithms of the core maintenance problem on edge-weighted graphs," *IEEE Access*, vol. 8, pp. 63872–63884, 2020.

[28] Y. Doytsher, B. Galon, and Y. Kanza, "Querying geo-social data by bridging spatial networks and social networks," in *Proc. 2nd ACM SIGSPATIAL Int. Workshop Location Based Social Netw. (LBSN)*, 2010, pp. 39–46.

[29] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2010, pp. 939–948.

[30] D.-N. Yang, Y.-L. Chen, W.-C. Lee, and M.-S. Chen, "On social-temporal group query with acquaintance constraint," 2011, *arXiv:1104.3219*. [Online]. Available: http://arxiv.org/abs/1104.3219

[31] Q. Fang, J. Sang, C. Xu, and Y. Rui, "Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning," *IEEE Trans. Multimedia*, vol. 16, no. 3, pp. 796–812, Apr. 2014.

[32] S. Liu, C. Jiang, Z. Lin, Y. Ding, R. Duan, and Z. Xu, "Identifying effective influencers based on trust for electronic word-of-mouth marketing: A domain-aware approach," *Inf. Sci.*, vol. 306, pp. 34–52, Jun. 2015.

[33] Y. Li, R. Chen, J. Xu, Q. Huang, H. Hu, and B. Choi, "Geo-social K-cover group queries for collaborative spatial computing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2729–2742, Oct. 2015.

[34] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "When engagement meets similarity: Efficient (k,r)-core computation on social networks," 2016, *arXiv:1611.03254*. [Online]. Available: http://arxiv.org/abs/1611.03254

[35] C. Kang, S. Kraus, C. Molinaro, F. Spezzano, and V. S. Subrahmanian, "Diffusion centrality: A paradigm to maximize spread in social networks," *Artif. Intell.*, vol. 239, pp. 70–96, Oct. 2016.

[36] Y. He, W. Li, L.-W. Chen, G. Forgues, X. Gui, S. Liang, and B. Hou, "A social search model for large scale social networks," 2020, *arXiv:2005.04356*. [Online]. Available: http://arxiv.org/abs/2005.04356

XIAOXU SONG received the master's degree in computer science from the Shenyang University of Technology, in 2011. He is currently pursuing the Ph.D. degree with Northeastern University, Shenyang. His main research interests include location-based social networks and social network analysis.

BIN WANG received the Ph.D. degree in computer science from Northeastern University, in 2008. He is currently an Associate Professor with the Computer System Institute, Northeastern University. His research interests include design and analysis of algorithms and queries processing over streaming data and distributed systems.

XIAOCHUN YANG (Member, IEEE) received the Ph.D. degree in computer science from Northeastern University, China, in 2001. She is currently a Professor with the Department of Computer Science, Northeastern University. Her research interests include data quality and data privacy. She is a member of ACM and the IEEE Computer Society. She is also a Senior Member of CCF.

JING QIN was born in Shenyang, China, in 1981. She is currently pursuing the Ph.D. degree with Northeastern University, China. Her research interests include recommendation systems and data mining.

LIANG ZHAO received the Ph.D. degree from the School of Computing, Edinburgh Napier University, in 2011. He is currently an Associate Professor with Shenyang Aerospace University, China. He has published more than 70 articles. His research interests include ITS, VANET, WMN, and SDN. He served as the Chair for several international conferences and workshops, including the IoT-Smart (Program Co-Chair), in 2015, the 2019 IEEE IUCC (Program Co-Chair), the 2019 IEEE Scalcom (Poster/Demo Co-Chair), the 2019 AI-Driven Network Workshop (Program Co-Chair), and the NGDN Workshop (Founder). He serves as a Guest Editor for the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, the *Journal of Computing* (Springer), and *Internet Technology Letters* (Wiley).

LIANQIANG NIU received the master's degree in computational mathematics from Jilin University, China, in 1986. He is currently a Professor with the Department of Computer Science, Shenyang University of Technology, China. His research interests include image processing and pattern recognition, information processing, and visualization.