Querying Intimate-Core Groups in Weighted Graphs

Dong Zheng*, Jianquan Liu†, Rong-Hua Li‡, Cigdem Aslay§, Yi-Cheng Chen††, Xin Huang*¶
*University of British Columbia, {zhdong, xin0}@cs.ubc.ca
¶Hong Kong Baptist University, xinhuang@comp.hkbu.edu.hk
†NEC Corporation, j-liu@ct.jp.nec.com
‡Shenzhen University, rhli@szu.edu.cn
§ISI Foundation, cigdem.aslay@isi.it
††Tamkang University, ycchen@mail.tku.edu.tw

***Abstract*—The semantic intimacy of relations in many real-world networks (e.g., social, biological, and communication networks) can be modeled by weighted edges in which the more semantically intimate relations between the nodes translate to smaller edge weights. Recently, the problem of *community search* that aims to find the cohesive groups containing a given set of query nodes has attracted a great deal of attention. However, the bulk of literature on *community search* problem assumes a simple unweighted input graph, ignoring how semantically intimate the nodes have in retrieved communities. The discovered communities may have a highly cohesive structure, while they perform poorly in the semantic of intimate connections.**

In this paper, we investigate a novel problem of Querying Intimate-Core Groups (QICG)**: given a weighted undirected graph G, a set of query nodes Q and a positive integer k, to find a connected subgraph of G in which each node has at least k neighbors, and the sum of weights on its edges is minimum among all such subgraphs. We show that the** QICG **problem is NP-hard. We develop efficient algorithms based on several practical heuristic strategies to enhance the retrieval efficiency. Extensive experiments are conducted on real-world datasets to evaluate efficiency and effectiveness of proposed algorithms. The results confirm that our intimate-core group model outperforms state-of-the-art models in weighted graphs.**

I. Introduction

Community detection, the task of identifying all communities in a given network, is a fundamental and well-studied problem [1], [2], [3], [4], [5], [6], [7]. Recently, a query-dependent variant of community detection problems, *community search* [8], has also attracted substantial interest. Given an input graph and a set of query nodes, community search problem aims to find a densely connected subgraph that contains given query nodes. Several papers have proposed various community search models based on different notions of dense subgraph structures, including quasi-clique [9], the densest subgraph [10], k-core [8], [11], [12], [13], and k-truss [14], [15].

The community search problem has many important real-world applications in broad areas such as content recommendation, finding groups in collaboration or protein-interaction networks, infectious disease control, and marketing [8]. Some typical applications include:

- **Tag recommendation.** A tag graph relates similar tags in a social-media platform: two tags are connected in the tag graph if they co-occur together. When a user uploads a photo to a photo-sharing portal with initial tag assignments, the system can recommend additional tags to the user based on the initial tags provided.
- **Finding groups.** A collaboration network encodes the co-authorship information between the scientists: two scientists are connected in the graph if they have co-authored at least one paper. Consider the scenario in which a group of scientists plan to apply for funding to explore new research directions. The chances of success of the application will be higher if they invite a set of colleagues such that each scientists is well-acquainted in this group, and even they have frequently worked together.
- **Infectious disease control.** If a person has come into contact with an infectious disease, her community of close contacts should be monitored, especially the individuals whom she frequently interacts.

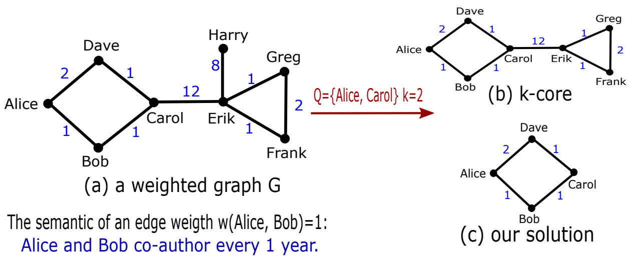


Fig. 1. An example intimate-core group query. (a) a given weighted graph G. (b) a k-core community retrieved by a general k-core algorithm without considering the weights in G. (c) a smaller community that has the minimum total weight retrieved by our approach. It can be confirmed that the community in (c) is semantically more intimate than the community in (b).

In the aforementioned applications, the standard community search models focus on the binary nature of the relationship between the nodes that is encoded via the (non-)existence of the edges in a graph. However, these relations can often be further quantified by the assignment of edge weights that better integrate and enhance the semantics of connections, hence, the communities. Thus, such graphs can instead be modeled with edge weights that capture the *semantic* intimacy of the nodes in which the more semantically intimate relations between the nodes translate to smaller edge weights. For instance, a good tag recommendation contains the tags that *frequently* co-occur

978-1-5090-4284-5/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSC.2017.80

with the initial tags provided by the user, where the interval of co-occurance of tags can be modeled as the edge weights in the tag graph. Similarly, the collaboration network can be modeled as a weighted graph in which the edge weights denote the time interval of collaborations: if two scientists collaborate frequently, the time interval of collaborations will be short, which translate to smaller edge weights. Thus, to improve the cohesive accuracy of the retrieved communities, one would want to ensure that the edges in the discovered communities have more semantic intimacy hence smaller edge weights as possible.

The bulk of literature on community search problem assume that the input graph is unweighted and focus only finding a cohesive structure without considering how semantically intimate the nodes in the retrieved community are. Moreover, the standard community search models admit several optimal solutions, some of which can be quite large, possibly due to the several irrelevant nodes they contain. To address this issue Sozio et al. [8] proposed a *constrained* version of the community search problem in which the size of the retrieved communities are subject to an upper bound. Similarly, Barbieri et al. [13] proposed an alternative formulation of the community search problem which aims to find the smallest-size solution among all the optimal solutions in an unweighted input graph so that the retrieved community is as cohesive as possible.

In this paper we drop the assumption that the input graph is unweighted and formulate the community search problem by taking into account the edge weights that denote the semantic intimacy between the nodes. We then study the problem of finding the intimate-core group of a given set Q of query nodes where we define the intimate-core group as the most semantically intimate community that contains Q. Our problem formulation takes as input a weighted undirected graph G, a set Q of query nodes, and a positive integer k, and aims to find a densely connected subgraph $H \subseteq G$ such that (*i*) H contains the query nodes, (*ii*) each node in H has at least k neighbors, and (*iii*) the sum of edge weights in H is the minimum among all the subgraphs that satisfy (*i*) and (*ii*).

Example 1. *Consider the collaboration network in Fig. 1(a). The weight of an edge between Alice and Bob is 1, indicating that Alice and Bob have co-authored every one year. Suppose we want to find an intimate-core group that contains the query nodes Alice and Carol. For $k = 2$, our solution will find the densely connected subgraph depicted in Fig. 1(c), in which every node co-authors with two other nodes and the sum of edge weights is equal to* 5.

On the other hand, the standard core decomposition approach that ignores the edge weights returns the k-core of G, depicted in Fig. 1(b): in this solution, several nodes have no co-authorship relation with the query nodes, e.g., Greg and Frank. Although Eric have co-authored with Carol, the interval of collaboration spans very long, every 12 *years. The sum of edge weights of this subgraph is equal to* 21. *Clearly, the retrieved community in Fig. 1(b) is semantically more intimate and cohesive than the community in Fig. 1(c).*

In our problem formulation, a good community should have a densely connected structure and small group weight where we define the weight of a group as the sum of its edge weights. In order to find densely connected groups that contain the query nodes, we use the minimum degree of nodes in the extracted groups to circumvent the free-rider issue. Since the standard definition of k-core does not necessarily induce a connected subgraph [16], we introduce "*connected* k-core", a dense and connected substructure in which each node has at least k neighbors. We then formulate intimate-core group model, a novel community search model, based on the definition of *connected* k-cores.

Contributions and roadmap. In this paper, we make the following contributions:

- We motivate the problem of Querying Intimate-Core Groups (QICG) in weighted graphs, in which the edge weights represent the semantic intimacy between nodes. We define a dense subgraph of connected k-core, and design an weight score function to measure intimate strength in groups. Based on this, we propose a novel group model called intimate-core group (ICG), and formulate the problem of QICG as finding ICGs (Section III)
- We formally demonstrate that QICG problem is NP-complete (Section IV).
- We devise an algorithmic framework that finds the intimate-core group for a given set of query nodes and a positive integer k. Our greedy algorithm ICG-S first identifies a maximal connected k-core as a candidate, then removes the node with the largest total weight on its incident edges at each iteration. To improve the efficiency of ICG-S, we also devise ICG-M, a variant of ICG-S that heuristically removes a batch of nodes at each iteration, reducing the total number of iterations required by ICG-S (Section V).
- We conduct extensive experiments on real-world weighted graphs and show the significant efficiency and effectiveness of our intimate-core group model and algorithms (Section VI).

We review the related work in Section II and conclude the paper in Section VII with a discussion of the future work.

II. Related Work

A. Community Search

Sozio et al. [8] are the first to define the combinatorial optimization formulation of the community search problem. They formulated the problem as the task of finding a connected subgraph that contains all the query nodes and maximizes the minimum degree. As noted by Sozio et al., one of the drawbacks of their formulation is that the retrieved communities can be arbitrarily large with several irrelevant nodes. In order to address this issue and improve the accuracy of the retrieved solutions, they also studied the integration of the distance and community size constraints. Following their formulation, many community search models have been studied in the literature based on different subgraph structures such as quasi-clique [9], k-core [12], [13], k-truss [14], [15], influential community [11], and query biased densest subgraph [10]. To improve the accuracy of the discovered communities, Barbieri et al. [13] proposed an alternative formulation of the community search problem that aims to find the *minimum* size community. They also proposed a precomputation based approach that exploits the core decomposition of the input graph to improve the efficiency of the computation. Cui et al. [9] designed a novel α-adjacency γ-quasi-k-clique model for finding overlapping communities. Cui et al. [12] also proposed an efficient local search algorithm for the restricted single-node query version of the problem. Huang et al. [14] propose a k-truss community model based on triangle adjacency for finding all the communities of a given query node. Li et al. [11] studied *influential* community model which reports the top-r communities with the highest influence scores over the entire graph. Wu et al. [10] and Huang et. al [15] independently proposed different models to address the free rider effect and avoid irrelevant nodes included in the communities. All the core-based community models discussed consider simple *unweighted* graphs which is a significant difference from our work.

B. Community Detection

Community detection aims to find all the communities in a graph. A typical approach for finding communities is to optimize the modularity measure [17]. In terms of community memberships, community detection falls into two major categories: non-overlapping community detection [1], [2], [3] and overlapping community detection [4], [5], [6], [7]. All these papers focus on static communities where the graphs are partitioned a priori. Community detection in dynamic graphs is instead addressed in [18]. An in-depth survey and performance comparison of the community detection algorithms can be found in [19].

C. Cohesive Subgraph Mining

Considerable work has been done on mining cohesive subgraph patterns, including clique [20], [21], [22], [23], quasi-clique [24], dense neighborhood graph [25], k-core [26], [16], and k-truss [27], [28], [29], [30]. Clique and quasi-clique enumeration methods include the classical algorithm [22], redundancy-aware clique enumeration [20], the external-memory H^*-graph algorithm [23], maximum clique computation using MapReduce [21], and optimal quasi-clique mining [24]. Various papers study core and truss decomposition in different settings, including in-memory algorithms [26], [27], [29], external-memory algorithms [16], [28], MapReduce [31] and probabilistic graphs [30]. Zhang et al. [29] and Huang et al. [14] design incremental algorithms for the efficient update of a k-truss with edge insertions / deletions. Wang et al. [25] studied a dense neighborhood graph based on common neighbors.

Notation	*Description*
$G = (V, E, w)$	Undirected weighted graph G
$w(e)$	The weight of edge $e \in E$
$n; m$	The number of nodes / edges of G
$N(v)$	The set of neighbors of v
$\deg_H(v)$	The degree of node v in subgraph $H \subseteq G$
d_{max}	The maximum node degree in G
$\delta(H)$	Coreness of subgraph $H \subseteq G$
$\delta(v)$	Coreness of node v
$\mathsf{f}_H(v)$	Node weight of v in subgraph $H \subseteq G$
$\mathsf{f}(H)$	Group weight of graph H

TABLE I
NOTATIONS USED IN THIS PAPER.

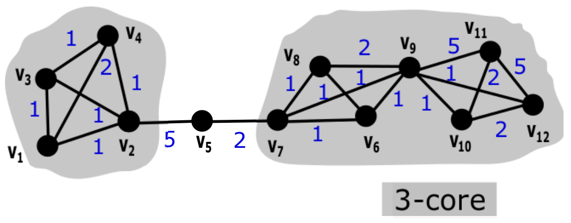


Fig. 2. An example 2-core graph G. Dashed region corresponds to the 3-core subgraph of G

III. Problem Statement

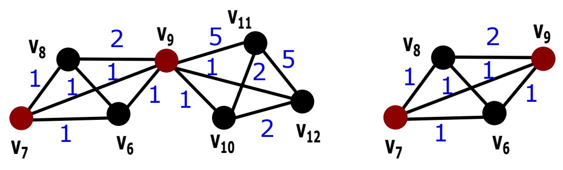
In this paper, we focus on a weighted undirected simple graph $G = (V, E, w)$, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $w : E \to \mathbb{R}^{\geq 0}$ is a weight function. Note that, for each edge $e = (u, v) \in E$, the weight $w(e) \geq 0$ denotes the semantic distance between the nodes u and v. Let $N(v) = \{u \in V : (v, u) \in E\}$ and $d(v) = |N(v)|$ respectively denote the set of neighbors and the degree of a node $v \in V$. Let $d_{max} = \max_{v \in V} d(v)$ denote the maximum node degree in G. A summary of the frequently used notations are depicted in Table I. Next we introduce the preliminary concepts required to formally define our problem.

k-core. A k-core is the maximal subgraph of G, in which each node has at least k neighbors in the subgraph.

Notice that, the basic definition of k-core does not necessarily induce a connected subgraph [16]. For instance, consider the weighted graph G in Fig. 2. The whole graph G is 2-core, since each node has degree of at least 2 while the 3-core subgraph of G depicted in the dashed region is disconnected with two components. This suggests that the basic k-core definition is not sufficient to define a densely-connected group. Thus, we first introduce *connected k-core*, a dense and connected substructure which we define by combining ideas from the k-core and connectivity concepts.

Definition 1 (Connected k-core)**.** *Given a subgraph $H \subseteq G$, H is a connected k-core iff every node has at least k neighbors in H, and there exists a path between any pair of nodes in H.*

Intuitively, a connected k-core of r nodes is a relaxed definition of r-clique, which allows a portion of edges to be missing from a complete graph of r nodes. Notice that,



(a) H_1 (b) H_2

Fig. 3. Querying intimate-core group for $Q = \{v_7, v_9\}$ and $k = 3$: the weight of H_1 and H_2 are respectively $\mathsf{f}(H_1) = 23$ and $\mathsf{f}(H_2) = 7$, thus, H_2 is the intimate-core group for the given query.

a $(k+1)$-clique is a connected k-core, but a connected k-core may not be a $(k+1)$-clique. In addition, a $(k+1)$-clique is the smallest connected k-core in terms of node and edge size.

Definition 2 (Coreness)**.** *The coreness of a subgraph $H \subseteq G$ is the minimum degree of a node in H, i.e., $\delta(H) = \min_{v \in V(H)}\{\deg(v)\}$. The coreness of a node $v \in V(G)$ is $\delta(v) = \max_{H \subseteq G \wedge v \in V(H)} \{\delta(H)\}$.*

Example 2. *Consider the graph G in Fig. 2. The coreness of G is $\delta(G) = \min_{v \in V(G)}\{\deg(v)\} = \deg(v_5) = 2$. The subgraph of G in dashed region in Fig. 2 is 3-core. In addition, for node v_7, the coreness of v has $\delta(v_7) = \max_{H \subseteq G \wedge v_3 \in V(H)} \{\delta(H)\} = 3$. There exists no 4-core in graph G, and the dashed region of 3-core is the subgraph of G with the maximum coreness.*

Definition 3 (Group Weight)**.** *Given a subgraph $H \subseteq G$, the group weight of H is defined as the sum of weights of all edges in H, denoted by $\mathsf{f}(H) = \sum_{e \in E(H)} w(e)$.*

Example 3. *Consider the subgraph $H_2 \subseteq G$ depicted in Fig. 3(b). The group weight of H_2 is obtained by summing the weights of all the edges in H_2, i.e., $\mathsf{f}(H_2) = \sum_{e \in E(H)} w(e) = 1+1+1+1+2+1 = 7$.*

Next we formally define the *intimate-core group* for a given weighted graph G, a set of query nodes Q, and a positive integer k.

Definition 4 (intimate-core group)**.** *Given a weighted graph G, a set of query nodes Q and a positive integer k, a subgraph $H \subseteq G$ is an* intimate-core group*, if H satisfies the following three conditions:*

- (1) Query Participation. *H contains all query nodes Q, i.e., $Q \subseteq V(H)$;*
- (2) Connected k-Core. *H is a connected k-core, $\forall v \in V(H)$, $\delta(v) \geq k$;*
- (3) Smallest Weight. *H is the subgraph with the smallest group weight among all subgraphs that satisfy conditions (1) and (2). That is, $\nexists H' \subseteq G$, such that $\mathsf{f}(H') < \mathsf{f}(H)$ and H' satisfies conditions (1) and (2).*

Example 4. *Consider again the weighted input graph G (Fig. 2). For a given set $Q = \{v_6, v_7\}$ of query nodes and $k = 3$, both H_1 and H_2, depicted in Fig. 3(a) and (b), are connected 3-cores containing Q. $\mathsf{f}(H_1) = \sum_{e \in E(H_1)} w(e) = 23$ and $\mathsf{f}(H_2) = \sum_{e \in E(H_1)} w(e) = 7$. However H_2 has the smallest group weight thus is the* intimate-core group *for the given values of Q and k.*

The problem of Querying Intimate-Core Groups studied in this paper is formulated as follows.

Problem 1 (Querying Intimate-Core Groups (QICG))**.** *Given a weighted undirected graph $G(V, E)$, a set of query nodes $Q = \{v_1, ..., v_{|Q|}\} \subseteq V$ and a number k, find the* intimate-core group *of Q.*

IV. Theoretical Analysis

In this section, we analyze the property of group weight function and the hardness of QICG problem.

Lemma 1. *Given a subgraph $H \subseteq G$, if H is a connected k-core, then $\mathsf{f}(H) \geq \frac{k(k+1)\lambda}{2}$ where λ is the minimum edge weight, i.e., $\min_{e \in E(H)} w(e) = \lambda$.*

Proof: Since H is a connected k-core, each node $v \in V(H)$ has at least k neighbors in H, i.e., $\deg_H(v) \geq k, \forall v \in V(H)$ and $|V(H)| \geq k+1$. Thus, the number of edges in H is $|E(H)| = \frac{\sum_{v \in V(H)} \deg(v)}{2} \geq \frac{k(k+1)}{2}$. As a result, $\mathsf{f}(H) = \sum_{e \in E(H)} w(e) \geq |E(H)| \times \min_{e \in E(H)} w(e) = \frac{k(k+1)\lambda}{2}$. ■

Theorem 1. *The* QICG *problem is NP-hard.*

Proof: We reduce the well-known NP-complete CLIQUE decision problem to the QICG problem. Given a graph $G = (V, E)$ and an integer k, the CLIQUE problem asks whether there exists a k-clique in G. Given an instance of CLIQUE problem with $G(V, E)$, we construct an instance of a weighted graph $G' = (V, E, w)$ for QICG as follows. For each edge $e \in E$, we assign the weight of 1 to e, i.e., $w(e) = 1$. Note that, for any subgraph $H \subseteq G'$, the graph weight of H is $\mathsf{f}(H) = \sum_{e \in E(H)} w(e) = |E(H)|$.

Next, we show that an instance of CLIQUE is a YES-instance iff for the corresponding instance of QICG has a connected k'-core H' containing Q with the graph weight $\mathsf{f}(H') <= \frac{k(k-1)}{2}$, where $k' = k-1$ and $Q = \emptyset$. The hardness follows from this.

$(\Leftarrow)$: Suppose a subgraph $H = (V(H), E(H)) \subseteq G$ is a YES-instance of CLIQUE problem, i.e., H is a k-clique. For each node $v \in V(H)$, $\deg(v) = k-1$ holds. H is also a connected $(k-1)$-core. In addition, $\mathsf{f}(H') = |E(H)| = \frac{k(k-1)}{2}$. Thus, as $Q = \emptyset \subseteq V(H')$, H' is also a YES-instance of our problem.

$(\Rightarrow)$: Suppose a subgraph $H' = (V(H), E(H), w)$ is a YES-instance of QICG problem. First, $\mathsf{f}(H') <= \frac{k(k-1)}{2}$. Second, by Lemma 1, we have $\mathsf{f}(H') \geq \frac{k(k-1) \times \min_{e \in E(H)}}{2} = \frac{k(k-1)}{2}$. $\mathsf{f}(H') = |E(H)| = \frac{k(k-1)}{2}$. In addition, since H has at least k nodes as a connected k-core, H is a complete graph of k vertices. Thus, H is also a YES-instance of CLIQUE problem. ■

V. Algorithms

In this section, we describe our algorithms for solving the QICG problem. We first describe an exact solution that straightforwardly enumerates all the possible subgraphs and returns the solution that has the minimum group weight. As the exact solution incurs a prohibitively high computation cost, we propose an efficient greedy algorithm that returns an approximate solution to the QICG problem.

A. An Exact Approach

A straightforward approach to solve the QICG problem is to compute the set of subgraphs that satisfy the *query participation* and *connected k-core* constraints and returning the one with the minimum weight as the intimate-core group solution. Obviously, this method is practically inefficient as the number of possible subgraphs that satisfy both the *query participation* and the *connected k-core* constraints can be exponentially large. Moreover, for each candidate subgraph, we also need to compute their group weights by scanning all the edges. Next we introduce our algorithm that operates by initially finding a maximal subgraph that satisfies the *query participation* and the *connected k-core* constraints followed by the iterative removal of the nodes that are incident to edges with larger weights to obtain the final approximate solution.

B. ICG-S*: A Greedy Method*

In this section, we introduce our greedy algorithm that finds an approximate solution by employing a greedy removal strategy.

Definition 5 (Node weight). *Given a subgraph $H \subseteq G$ and a node $v \in V(H)$, the node weight of v in graph H is defined as the sum of weights of its incident edges in H, denoted by $\mathsf{f}_H(v) = \sum_{u \in V(H) \cap N(v)} w((v,u))$.*

Based on the notion of node weight, we propose a greedy algorithm, called ICG-S, to find a intimate-core group. ICG-S algorithm first identifies a maximal connected k-core as a candidate. Then, the algorithm iteratively deletes the nodes with the largest node weights from the candidate graph. To maintain the remaining graph as a connected k-core, the nodes having degree less than k are also deleted until the remaining graph does not have a connected k-core containing Q. The algorithm terminates by returning the connected k-core that has the minimum group weight among all the possible candidate groups.

The detailed description of the algorithm ICG-S is depicted in Algorithm 1. The algorithm takes as input a weighted undirected graph G, a set of query nodes Q, and a positive integer k. Initially the algorithm identifies the maximal connected k-core of G: the nodes that have degree less than k and their incident edges are iteratively deleted from G (lines 1-4). Then, the algorithm checks whether the set Q of query nodes are all contained in a connected component of G(lines 5-8). If the query nodes in Q are not connected in G, the algorithm terminates with an empty graph (line 6); otherwise finds the maximal connected k-core denoted by G_0 (line 8). Afterwards, the algorithm starts iteratively shrinking the candidate G_l where $l = 0$ into an answer with a small group weight (lines 10-14): the algorithm first computes all the node weights in G_l then selects a non-query node u^* with the largest node weight, i.e., $u^* \leftarrow \arg\max_{u \in V(G_l)-V_q} \mathsf{f}_{G_l}(u)$ (line 11). Then, node u^* and all its incident edges are deleted from graph G_l (line 12). After the deletion of u^*, the neighbors of u^* may have degree less than k. Thus, to maintain the remaining graph G_l as a connected k-core, we iteratively remove the nodes that have degree less than k from G_l (line 13). This procedure is repeated until G_l is not a connected k-core containing Q. Finally, the candidate solution G_{l-1} that has the minimum group weight is returned (line 15).

Algorithm 1 ICG-S (G, Q, k)

Input: A weighted graph $G = (V, E, w)$, a set of query nodes Q , a positive integer k.
Output: An approximate intimate-core group of Q.

```
1: while $\exists v \in V$ with $\deg_G(v) < k$ do
2:     for $u \in N(v)$ do
3:         $\deg_G(u) \leftarrow \deg_G(u) - 1$;
4:     Delete node $v$ and its incident edges from $G$;
5: if the set $Q$ of query nodes is disconnected in the maximal
   $k$-core, i.e., the current graph $G$ then
6:     return $\emptyset$;
7: else
8:     $G_0 \leftarrow$ maximal connected $k$-core of $G$ that contains
       $Q$;
9:     $l \leftarrow 0$;
10: while $Q$ is connected in $G_l$ do
11:     Select a non-query node $u^*$ with the largest weight
        $\mathsf{f}_{G_l}(u^*)$, i.e., $u^* \leftarrow \arg\max_{u \in V(G_l)-V_q} \mathsf{f}_{G_l}(u)$;
12:     Delete the node $u^*$ and its incident edges from $G_l$;
13:     Iteratively delete a node $u$ with $\deg_{G_l}(u) < k$ and its
        incident edges from $G_l$;
14:     $l \leftarrow l + 1$;
15: return $G_{l-1}$;
```

Example 5. *We apply Algorithm 1 on the weighted graph G in Figure 2 with query $Q = \{v_7, v_9\}$ and $k = 3$. First, the algorithm detects the connected 3-core of G_0 as the subgraph H_1 shown in Figure 3(a). Next, we compute the node weights and detect node v_{11} as the node with the maximum weight $\mathsf{f}_{H_1}(v_{11}) = 5 + 5 + 2 = 12$. Then, we delete v_{11} from G_0. Meanwhile, other two nodes v_{10} and v_{12} are also deleted from G_0, since they have degree less than 3. Finally, the algorithm finds the* intimate-core group *H_2 with the minimum group weight in Figure 3(b), which is the optimal solution in G.*

In the following, we analyze the time and space complexity of Algorithm 1.

Theorem 2. *The method* ICG-S *in Algorithm 1 takes $O(tm)$ time and $O(m)$ space, where $t \leq n$.*

Proof: The time cost of Algorithm 1 is mainly consisted of two parts: node weight computation and connected k-core maintenance. First, consider the node weight computation:

computing the node weights of all the nodes takes $O(m)$ time. Finding the node with the maximum node weight takes $O(n)$ time. Since the algorithm runs in t iterations, this step takes $O(tm)$. Now, consider the identification and the maintenance of connected k-core. Identifying and maintaining a series of k-core candidate graph $\{G_0, ..., G_{l-1}\}$ takes $O(m)$ time in each iteration. However, checking whether each candidate graph is connected takes $O(m)$ time in each iteration. Thus, the total time cost of this step is $O(tm)$. As a result, the overall time complexity of Algorithm 1 is $O(tm)$.

Now, we analyze the space complexity. For each node $v \in G_i$, we keep the degree $\deg_{G_i}(v)$ and node weight $\mathsf{f}_{G_i}(v)$ which takes $O(n)$ space. Note that, for all candidate graphs $\{G_0, ..., G_l\}$, we do not store their nodes and edges. Instead, we keep the sequence of removed nodes from G_0 by assigning remarks of candidate graphs, which takes $O(m)$ space in all. Therefore, the space complexity is $O(m)$, due to the assumption $n \leq m$. ■

C. ICG-M: *A Fast Greedy Method*

Although ICG-S (Algorithm 1) is much more efficient than the brute-force exact algorithm, it takes $O(tm)$ time for each query, which is still inefficient for large graphs. Moreover, since Algorithm 1 deletes only one node from the graph at each iteration, the number of iterations can render quite large. Thus, to speed up Algorithm 1, we invoke the technique of bulk deletion [32] which deletes multiple nodes from the current candidate graph in each iteration.

Specifically, assume that G_i is the current candidate graph and let S denote the set of nodes that have the largest node weight in G_i. Let $|S| = \gamma|V(G_i)|$ for a $\gamma \in (0,1)$. In the node removal step at each iteration, our goal is to remove a set of nodes S as opposed to one v at a time (lines 11 and 12 in Algorithm 1). Our new approach follows the framework of Algorithm 1 by simply replacing the single node removal with this multiple nodes removal strategy. We denoted the new method as ICG-M.

Now, let's figure out how many iterations ICG-M takes. Initially ICG-M finds a candidate graph of maximal connected k-core as G_0, which contains at most n vertices in total, i.e., $|V(G_0)| \leq |V(G)| = n$. Then, in the following process, we iteratively shrink the candidate graph by removing a set of nodes S. At the i-th iteration, for current candidate graph G_i, $|S| = \gamma|V(G_i)|$ nodes will be deleted from G_i. This indicates that there remains at most $(1-\gamma)|V(G_i)|$ nodes in next candidate graph G_{i+1}, i.e., $|V(G_{i+1})| \leq (1-\gamma)|V(G_i)|$. Suppose the algorithm needs t' iterations to terminate, we have $1 \leq |V(G_{t'})| \leq (1-\gamma)|V(G_{t-1})| \leq ... \leq (1-\gamma)^{t'}|V(G_0)| \leq (1-\gamma)^{t'}n$. As a result, $t' \leq \lceil \log_{\frac{1}{1-\gamma}} n \rceil$ holds. Thus, ICG-M achieves the time complexity of $O(m \log_{\frac{1}{1-\gamma}} n)$, where $\gamma \in (0,1)$.

Theorem 3. *The method* ICG-M *that uses the bulk deletion strategy takes $O(m \log_{\frac{1}{1-\gamma}} n)$ time and $O(m)$ space, where $\gamma \in (0,1)$.*

TABLE II
NETWORK STATISTICS

Network	$\|V\|$	$\|E\|$	d_{max}	δ_{max}	**Avg. Edge Weight**
Fruit-Fly	3751	3,692	27	4	4.416
DBLP	996,674	3,966,007	16330	286	0.867

Proof: The proof is similar with Theorem 2, which is omitted. ■

VI. Performance Studies

In this section, we conduct extensive experiments over real-world datasets to evaluate the efficiency and the effectiveness of the proposed algorithms.

A. *Datasets and Experimental Setup*

Datasets. We use two real-world weighted networks in our experiments: Fruity-Fly and DBLP. Fruit-Fly is a protein-protein interaction (PPI) network, obtained by integrating data from the BioGRID[1] database and data from the STRING database. The graph contains 3,751 nodes and 3,692 edges, where the weight of an edge corresponds to the confidence that the interaction may not exist: the smaller the weight of an edge is, the larger probability of interaction. DBLP is a collaboration network constructed from the DBLP dataset[2]. A node represents an author and an edge is added between two authors if they have co-authored a paper at least once. The weight of an edge (u,v) is defined as $\frac{1}{\mathsf{freq}(u,v)}$, where $\mathsf{freq}(u,v)$ is the number of times that u and v have co-authored. The smaller the weight of an edge is, the more frequent the incident nodes have ever co-authored. The network contains 996,674 nodes and 3,966,007 edges. The detailed statistics of datasets are summarized in Table II, where d_{max} is the maximum degree of a node in G and δ_{max} is the maximum coreness of a node in G.

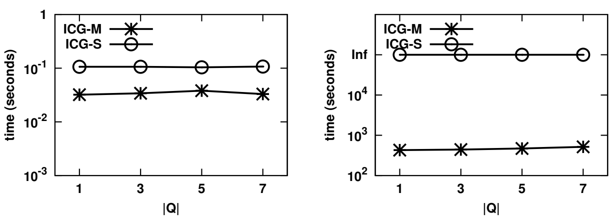
Comparison Methods. To evaluate the efficiency and effectiveness of finding intimate-core groups methods, we compare the following algorithms:

1) ICG-S Algorithm 1 that deletes the node with the largest node weight at each iteration,
2) ICG-M Variant of Algorithm 1 that uses a bulk deleting strategy at each iteration,
3) k-core The standard core based method that returns the maximal connected k-core which contains all the query nodes.

All algorithms are implemented in Java, and all the experiments are conducted on a Windows System with Intel Core i5-2450M CPU (2.50GHZ) and 12GB main memory.

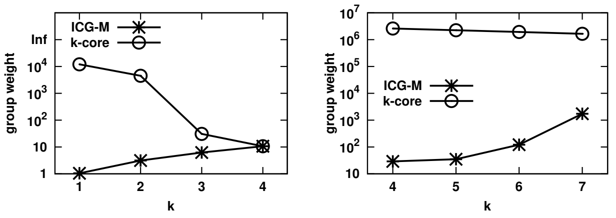
Queries. For each dataset, we randomly generate 100 sets of queries. Unless otherwise specified, each query set contains 2 nodes. We set $k = 2$, and $k = 5$ for Fruit-Fly and DBLP datasets respectively by default.

[1]http://thebiogrid.org/
[2]http://dblp.uni-trier.de/xml/



(a) Fruit-Fly (b) DBLP

Fig. 4. Running time comparison (in seconds) for varying query set sizes.



(a) Fruit-Fly (b) DBLP

Fig. 5. Effectiveness comparsion of k-core and ICG-M on Fruit-Fly and DBLP graphs in terms of group weight.

B. Efficiency Evaluation

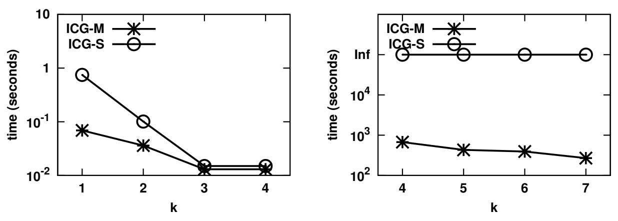
We first evaluate the running time of ICG-S and ICG-M for different values of $|Q|$ and k on Friut-Fly and DBLP networks. We define the running time of a query as infinite if the algorithm can not terminate in 2 hours.

Varying $|Q|$. To evaluate the performance of the algorithms for different query set sizes, we vary $|Q|$ from 1 to 7. The running time results are shown in Fig. 4. For Fruit-Fly dataset, both algorithms achieve stable efficiency performance with the increasing values of $|Q|$. On the other hand for DBLP, ICG-S was not able to terminate within 2 hours thus is regarded as infinite. Results on both datasets show that ICG-M significantly outperforms ICG-S thanks to the bulk deletion strategy that requires less number of iterations. These results verify our time complexity analysis.

Varying k. To evaluate the performance of the algorithms for different query set sizes, we vary k from 1 to 4 on the Fruit-Fly dataset and from 3 to 7 on the DBLP dataset. As we can see, in Fig. 6 (a), both methods take less time when k increases. In general ICG-M always has better run time than ICG-S.

C. Effectiveness Evaluation

In this experiment, we evaluate the effectiveness of intimate-core group model in weighted graphs. We compare our method ICG-M with k-core method on Fruit-Fly and DBLP datasets. We report the group weights of results generated by two methods in Fig. 5. We first vary k from 1 to 4 on the small Friut-Fly graph. In Fig. 5(a), ICG-M achieve much smaller intimate-core group, which indicates our intimate-core group model can find a cohesive group with the small group weight. We also vary k from 3 to 7 on the large DBLP graph. The



(a) Fruit-Fly (b) DBLP

Fig. 6. Efficiency comparsion of ICG-M and k-core on Fruit-Fly and DBLP graphs in terms of running time (in seconds). The default $|Q| = 2$.

results shown in Fig. 5 (b) also verify the superiority of our intimate-core group model.

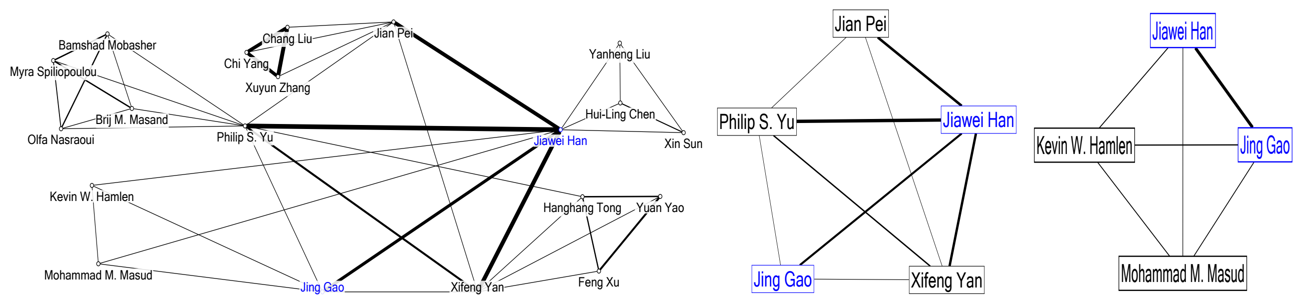
D. Case Study on DBLP Network

We test our ICG-M algorithm on the DBLP network to find intimate-core groups. We use the query $Q = \{$"Jiawei Han", "Jing Gao"$\}$ and set $k = 3$. Figure 7(a) depicts[3] a connected 3-core of G containing both query nodes. It has 20 nodes, 43 edges, and a group weight of 16.6. As we can see, most of authors are loosely connected to query authors, and most of edges are associated with large weights. Our method ICG-M removes the nodes that are loosely connected with query nodes, and finds an intimate-core group for Q shown in Figure 7(b): this community has 5 nodes, 9 edges, edge density of 0.9, and group weight of 3.2, which is far less than the group weight of the graph in Figure 7(a). In addition, we also consider a variant of k-core model to find the community with the minimum size in unweighted graphs [13]. We apply this method by assigning equal weights to all edges in the DBLP network. Figure 7(c) shows one of the minimum 3-core groups that contain Q. It contains 4 nodes, 4 edges and group weight of 3.4, which is semantically less intimate than our intimate-core group in Figure 7(b). This is due to the fact that their method does not consider the edge weights that reflect the semantic intimacy, and focus solely on minimizing the size of the community. Overall, the results clearly shows the superiority of intimate-core group model, which can detect densely connected and semantically intimate groups in weighted graphs.

VII. Conclusion

In this paper, we study the problem of querying intimate-core groups in weighted graph. We propose a novel model of intimate-core group based on the concepts of connected k-core and group weight. We show that the problem of finding intimate-core groups is NP-hard and devise an algorithmic framework that efficient finds the intimate-core groups. Our greedy algorithm ICG-S, after initially identifying a maximal connected k-core as a candidate, removes the node with the largest total weight on its incident edges at each iteration. We also devise a more efficient algorithm ICG-M that heuristically removes a batch of nodes to reduce the total number of

[3]We do not display an entire graph of the maximal connected 3-core in Figure 7(a), due to the large number of nodes it contains.



(a) Connected 3-core (b) ICG-M (c) Minimum k-core

Fig. 7. A case study of intimate-core group search with query nodes $Q = \{$"Jiawei Han", "Jing Gao"$\}$. In the plots, a wider line indicates a smaller weight on the edge.

iterations required by ICG-S. We conduct extensive experiments on real-world weighted networks, and demonstrate the effectiveness and efficiency of our proposed models and algorithms. Our work opens several interesting avenues for future work. Some examples include developing I/O-efficient algorithms for finding intimate-core groups over massive weighted graphs and investigating intimate-core groups on location-based social networks.

Acknowledgment

We would like to thank all the reviewers for their insightful and valuable comments. Yi-Cheng Chen was supported by the Ministry of Science and Technology, Taiwan, Project, No. MOST 104-2221-E-032-037-MY2. This work was partially done by the author Xin Huang while at Hong Kong Baptist University. Xin Huang is the corresponding author of this paper.

References

[1] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.
[2] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.
[3] Zhirong Yang, Tele Hao, Onur Dikmen, Xi Chen, and Erkki Oja. Clustering by nonnegative matrix factorization using graph random walk. In *NIPS*, pages 1079–1087, 2012.
[4] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.
[5] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.
[6] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Comput. Surv.*, 45(4):43, 2013.
[7] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596, 2013.
[8] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *KDD*, pages 939–948, 2010.
[9] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. Online search of overlapping communities. In *SIGMOD*, pages 277–288, 2013.
[10] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. Robust local community detection: On free rider effect and its elimination. *PVLDB*, 8(7), 2015.
[11] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. Influential community search in large networks. *PVLDB*, 8(5), 2015.
[12] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. Local search of communities in large graphs. In *SIGMOD*, pages 991–1002, 2014.
[13] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. Efficient and effective community search. *Data Mining and Knowledge Discovery*, 29(5):1406–1433, 2015.
[14] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. Querying k-truss community in large and dynamic graphs. In *SIGMOD*, pages 1311–1322, 2014.
[15] Xin Huang, Laks VS Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. Approximate closest community search in networks. *PVLDB*, 9(4):276–287, 2015.
[16] James Cheng, Yiping Ke, Shumo Chu, and M. Tamer Özsu. Efficient core decomposition in massive networks. In *ICDE*, pages 51–62, 2011.
[17] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.
[18] Jierui Xie, Mingming Chen, and Boleslaw K Szymanski. Labelrankt: Incremental community detection in dynamic networks via label propagation. In *Proceedings of the Workshop on Dynamic Networks Management and Mining*, pages 25–32, 2013.
[19] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.
[20] Jia Wang, James Cheng, and Ada Wai-Chee Fu. Redundancy-aware maximal cliques. In *KDD*, pages 122–130, 2013.
[21] Jingen Xiang, Cong Guo, and Ashraf Aboulnaga. Scalable maximum clique computation using mapreduce. In *ICDE*, pages 74–85, 2013.
[22] Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.
[23] James Cheng, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. Finding maximal cliques in massive networks by h*-graph. In *SIGMOD*, pages 447–458, 2010.
[24] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *KDD*, pages 104–112, 2013.
[25] Nan Wang, Jingbo Zhang, Kian-Lee Tan, and Anthony K. H. Tung. On triangulation-based dense neighborhood graphs discovery. *PVLDB*, 4(2):58–68, 2010.
[26] V. Batagelj and M. Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.
[27] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. Technical report, National Security Agency, 2008.
[28] Jia Wang and James Cheng. Truss decomposition in massive networks. *PVLDB*, 5(9):812–823, 2012.
[29] Yang Zhang and Srinivasan Parthasarathy. Extracting analyzing and visualizing triangle k-core motifs within networks. In *ICDE*, pages 1049–1060, 2012.
[30] Xin Huang, Wei Lu, and Laks V. S. Lakshmanan. Truss decomposition of probabilistic graphs: Semantics and algorithms. In *SIGMOD*, pages 77–90, 2016.
[31] Jonathan Cohen. Graph twiddling in a mapreduce world. *Computing in Science and Engineering*, 11(4):29–41, 2009.
[32] Xin Huang and Laks VS Lakshmanan. Attribute truss community search. *arXiv preprint arXiv:1609.00090*, 2016.