


Query Optimal k-Plex Based Community in Graphs

Yue Wang¹  · Xun Jian¹ · Zhenhua Yang² · Jia Li²

Received: 24 October 2017 / Accepted: 31 October 2017 / Published online: 13 November 2017
© The Author(s) 2017, corrected publication 11/2017. This article is an open access publication

Abstract Community search problem, which is to find good communities given a set of query nodes in a graph, has attracted increasing research interest recently. Though various measurement models have been proposed to define and solve community search problem. Few of them could define a community concisely and have good quality of query results. They either involve additional constraints for modeling communities, such as size and diameter, or suffer from the free rider effect, i.e., include irrelevant subgraphs. In this paper, we propose a new k-plex based community model for community search. We show that our model not only is simple and clear, but also meets with basic requirements of defining a community search problem. We formulate the maximum k-plex community query (MCKPQ) problem, that is, given a set of query nodes Q , searching for optimal k-plex containing Q . We prove that MCKPQ is NP-hard, and it is hard to approximate in any constant factor. We first give exact solutions. Then, we propose an efficient branch-and-bound (B&B) method and design an effective upper bound function and a pruning strategy. Furthermore, we optimize the basic B&B by fast

candidate generation. We also give a fast heuristic solution, which produces high-quality results in practice. The effectiveness of our model of community and the efficiency of our methods are verified by elaborate experiments.

Keywords Community search · Graph algorithm · K-plex

1 Introduction

Community, which is defined as set of nodes densely connected internally, is considered as an important structure in networks and plays a significant role in graph mining. *Community detection* is a task to find communities for a graph, and it has many applications in different fields, such as: (1) mining sets of highly correlated stocks [27]; (2) detecting DNA motif in bioinformatics [25]; (3) finding Web sites communities sharing similar topics [23]. Numerous works, which are based on different community models, have been developed to detect communities [13, 32].

Since nowadays graph data become large and dynamic, much research attention has been transferred from the community detection problem to the *community query problem* [10, 11, 17, 28]. Unlike community detection, which enumerates all communities, community query aims to find the communities that contain a set of query nodes Q .

Compared with community detection, community search by query nodes could avoid: (a) the time consumed to find all communities of entire large graphs; (b) the inflexible global parameter for the community criterion; (c) the difficulty to deal with the dynamically evolving graphs [10]. In addition, community search has many applications in real-life networks on its own for query specific nodes:

✉ Yue Wang
ywangby@connect.ust.hk

Xun Jian
xjian@connect.ust.hk

Zhenhua Yang
silence.yang@huawei.com

Jia Li
lijia@huawei.com

¹ Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

² Huawei Technologies Co. Ltd., Xi'an, China

- *Social group finding* In an event-based social networks such as Meetup,¹ a common task is to personally search for a group with strong social ties among people, to organize interesting activities.
- *Friends recommendation* In popular social network applications, e.g., Facebook,² Twitter,³ and Instagram,⁴ community search could help predict potential friendship relationships for users, especially for new users with few friends.
- *Frequent pattern mining* When transactions and items are modeled as a bipartite graph, then dense communities group items which are bought together frequently in different transactions. Given some specific items, community search would help identifying frequent item sets and further benefiting marketing.

It is not straightforward to define what is a community and to formulate community search problem precisely. There are several aspects to be concerned, among which the most important are how to measure the goodness of a community and how to define it. Although there is no standard answer, we summarize common requirements that a formulated community search problem should meet:

1. *Cohesive* A community should be cohesive. Each member within it should be familiar with others. Currently, many cohesive measurements have been proposed to ensure the cohesiveness of a community, such as subgraph density [2, 9], minimum degree [11], and subgraph diameter [28].
2. *Number of query nodes* When $|Q| = 0$, community query is identical to community detection. Some works limit $|Q| = 1$, a single query node, which is reasonable but not general in all cases. It is required $|Q| \geq 1$ sometimes. For example, Alice and Bob are holding a party, and they want to invite friends who have strong social ties with both of them.
3. *Avoid free rider effect* Lots of community measurements lead to free rider effect [30], which means including unrelated subgraphs in query results and making communities impure. Thus, it is required that the query results should not have the free rider effect.
4. *Connectivity* Suppose there is a community H for the community query, then not only H should contain Q , but also H is supposed to be connected. However, the

latter requirement is not always guaranteed in different models.

5. *Size of community* In some scenarios, we are interested in not only cohesive communities given a query, but also those with size bound. As shown in [21], an activity may not be triggered on if the size of social group is less than the activity's lower bound for the number of participants. Sozio and Gionis [28] also study the problem of querying the optimal community with an upper bound of the size.
6. *Few parameters* In spite of above constraints, as indicated in [17], definitions of problems which are simple to formulate have few parameters are more preferable.

Currently, several models of community have been proposed to solve community query problem [4, 11, 17, 18, 28, 31]. However, none of them could satisfy the above requirements simultaneously, which means that more constraints would be added to their proposed models to cover missing requirement, and making the problems more complicated. Take k-truss model [17] as an example, Huang et al. [17] use k-truss property to ensure community cohesiveness and use edge connectivity to ensure the community connectivity. K-truss model is further improved by restricting community diameter, and avoiding free rider effect in [18]. k-core based (minimum degree) model is also used to define community in [4, 11, 22, 28]. Sozio and Gionis [28] first introduce minimum degree as the goodness function of community and give a linear time algorithm for unbounded size problem. Cui et al. [11] improve the unbounded size version using the local search strategy, but leave the problem with bounded size unsolved. Sozio and Gionis [28] also add the bounded diameter and the size constraint to keep community small. Cui et al [10] introduce a quasi-clique based community to discover overlapping communities for single query node, but include too many parameters and make the problem hard to set appropriate parameters, as pointed out in [17]. To address the free rider effect, which is common in most community models, Wu et al. [31] propose a query-density based model. In [31], the random walk is used to measure the density of each node from query nodes, and the problem is to find a community which maximizes the query-based density. However, the result communities have no size guarantee in this measurement.

To address the above problems, in this paper, we proposed a novel k-plex based community model for query communities. K-plex, first introduced in [26], is one kind of relaxation of a k-clique. A graph H is called a *k-plex* if for each node $v \in H$, v has at least $|H| - k$ neighbors. In other words, each member could have missed at most k non-neighbors in H . Clique is a special case of k-plex when $k = 1$.

The reason of that we model a community as a k-plex is it can meet all above requirements for the community

¹ <https://www.meetup.com/>.

² <https://www.facebook.com/>.

³ <https://twitter.com/>.

⁴ <https://www.instagram.com/>.

search problem, that is, a k-plex community H has good cohesive structure, bounded diameter, upper bound size guarantee. At the same time, the connectivity of H is guaranteed as H becomes large. The overview properties of a k-plex H are listed as below:

1. The diameter of H is bounded by k ;
2. the lower bound of the density of H is $\frac{|H|-k}{2}$, which makes community cohesive with its size increasing;
3. any k-plex larger than $k + 1$ is connected;
4. there is an upper bound for $|H|$;
5. when H is optimal (defined in Section 2) for query nodes, it can avoid free rider effect;
6. since a k-plex must be a $(k + 1)$ -plex, k-plex communities could form a hierarchy structure for query nodes, by querying with increasing k .

The detailed analysis would be shown in Sect. 2. In spite of listed properties, querying optimal k-plex community only needs specifying Q and k , making issuing a query as easy as possible.

Although the description of querying k-plex community problem is simple and clear, we show it is NP-hard. In addition, it is even NP-hard to approximate with $O(n)$ factor in polynomial time.

Thus, to search k-plex community efficiently, we first introduce two enumeration methods to get the exact solution, while the later can prevent repeated enumeration. Next, we come up with an efficient branch-and-bound framework. Furthermore, we reduce the search space by carefully defining an upper bound function. In addition, we also introduce *global pruning* and *local pruning* techniques to filter out unqualified neighbors. Then, we present two optimization methods, *partial branching* and *incremental generation*, to accelerate the candidate generation processes. In the end, we present heuristic solutions which can generate results fast with good quality in practice.

The rest of the paper is organized as follows. We give the problem definition in Sect. 2. We present baseline solutions in Sect. 3. In Sect. 4, we propose an efficient branch-and-bound framework. In Sect. 5, we introduce two optimization techniques to accelerate the B&B framework. In Sect. 6, we discuss a heuristic solution. Experiments are reported in Sect. 7. We discuss related works in Sect. 8 and conclusion in Sect. 9.

2 Problem Statement

In this section, we give some preliminaries and formulate our k-plex based community problem, and then we compare our model with current ones and present the analysis of our proposed problems.

2.1 Problem Definition

In this paper, we consider simple undirected graphs $G(V, E)$ which have no weight on nodes and edges. Let $n = |V(G)|$ and $m = |E(G)|$. We denote the neighbors of a node $v \in V$ by $N(v)$ so the degree of v is $deg_G(v) = |N(v)|$. For any $H \subseteq V$, the subgraph induced by H is denoted as $G[H]$ with nodes $V(G[H]) = H$ and edges $E(H) = \{(v_1, v_2) | v_1, v_2 \in H, (v_1, v_2) \in E(G)\}$. Sometimes we replace $G[H]$ by H if the context is clear (Table 1).

As a relaxation of clique, *k-plex* was first introduced in [26]. We now give the definition of *connected k-plex*.

Definition 1 (*Connected k-plex*) Given a graph G and a constant k , a subgraph $G[H]$ is said to be a connected k-plex if $G[H]$ is connected and for any $v \in H$, $deg_{G[H]}(v) \geq |H| - k$.

Lemma 1 *If G is a k-plex. Then, any vertex-induced subgraph from G is also a k-plex.*

Compared with *k-core* model, whose lower bound of minimum degree is a constant k , the minimum degree of a k-plex is increased with its cardinality. The larger of a k-plex, the more cohesive it is.

However, a k-plex defined by a single k would form different structural subgraphs. Consider $k = 1$, which is the minimum k we could provide, then both an edge and a clique could satisfy 1-plex property. Clearly the former one is less concerned than the latter. So we define the k-plex community problem with size constraint.

Definition 2 (*Connected k-plex query*) Given a graph G , a set of query nodes $Q \subseteq G$, a constant k and size constraint c , find a subgraph $G[H]$, such that:

1. $Q \subseteq H \subseteq V(G)$ and $G[H]$ is a connected k-plex.
2. $|H|$ is no less than c .

The results of *CKPQ* could be exponential. Consider a clique of size n , set an arbitrary single query node q , $k = 1$ and $c = 1$, then there would be 2^{n-1} candidate results (each subgraph whose size larger than 1 containing q would be a

Table 1 Notations and symbols

$G(V, E)$	A simple graph with $ V $ nodes and $ E $ edges
$G[H]$	The subgraph induced by a set of nodes $H \subseteq V$
Q	A set of query nodes $Q \subseteq V$
$deg_G(v)$	The degree of node v in graph G
$\delta(G)$	The minimum degree of nodes in $V(G)$
$N_G(q)$	Neighbors of node q in graph G
$\overline{N_G(v)}$	Non-neighbors of node v in G

feasible solution). Now we give the optimized version of k -plex community search problem.

Definition 3 (*Maximize connected k -plex query*) Given a graph G , k , a set of query nodes $Q \in G$ find a subgraph $G[H]$ whose size is maximized among all the solutions to CKPQ.

Theorem 1 *For each $q \in Q$, the maximized connected k -plex consisting of Q must be one of maximal k -plex of q .*

Proof Clearly the optimal solution H which consists of Q must be maximal, otherwise we can add extra nodes to current solution to make it larger, which results in better solution and contradicts with that H is optimal. Next, $q \in Q$, $Q \subset H$ implies H is maximal k -plex of q . \square

2.2 Compare with k -core Based Community

There are already some works about searching community with minimum degree guarantee [11, 28]. We make a comparison with k -plex community here and show what is the difference. For a community H , previous works mainly focus on make $\delta(H)$ large, where $\delta(H)$ defines the lower bound of neighbors for each node in $|H|$. However, k -plex defines upper bound of the number of each node would miss. That results in the larger H is, the more neighbors each node has. As a result, k -plex community is “denser.”

Definition 4 (*k -core*) Given graph $G(V, E)$, a subgraph $G[H]$ is called k -core or a core of order k iff $\forall v \in H, \deg_H(v) \geq k$ and H is a maximum subgraph with this property.

Without size bound, deciding whether there exists a H such that $\delta(H) \geq k$ can be done in linear time, shown in [11, 28]. However, with size bound, the problem becomes NP-hard. Authors of [11] provided efficient solution for the unbounded version and proposed bounded problem unsolved, shown below.

Definition 5 (*mCST [11]*) Given a graph $G(V, E)$ and a query node $v_0 \in V$ and a constant k , find $H \subset V$ such that (1) $\delta(G[H]) \geq k$; (2) $G[H]$ is connected; (3) $|H|$ is minimized.

Note that CKPQ(Q, k, c) is identical to decision version of mCST, by setting $|Q| = 1, k' = c - k, c' = c$. In this paper, we only focus on the optimization problem MCKPQ, since the solution of MCKPQ can help solving the CKPQ problem.

2.3 Hardness Results

In this section, we show the decision version of MCKPQ, CKPQ is NP-complete. We further show that MCKPQ is also hard to approximate in any constant factor.

Theorem 2 *The CKPQ Problem is NP-Complete for any constant k and $|Q| \geq 1$.*

Proof Clearly, given Q, k, c and a candidate subgraph H , we could test whether H is a k -plex with size larger than c and contains Q in polynomial time.

Next, we reduce from k -plex problem, which is shown NP-complete for any positive integer k in [3]. Given a graph $G(V, E)$ and k , the k -plex problem is to decide whether there exists a k -plex of size c in G . We now construct an instance of CKPQ by constructing a new graph $G'(V', E')$ as follows: we create arbitrary set of nodes Q and add them to V , so $V(G') = Q + V(G)$, for each node v in Q , we connected v to all the other nodes, i.e., $E(G') = E(G) \cup \{(v_1, v_2) | v_1 \in Q, v_2 \in V', v_2 \neq v_1\}$, obviously $G'[Q]$ is a clique, and we finish construction by setting $c' = c + |Q|$ and Q as query nodes. We show that k -plex problem is a Yes-instance iff the decision version of MCKPQ is a Yes-instance. Suppose subgraph $G[H]$ is a solution of k -plex problem, then $H' = G'[H \cup Q]$ is a solution of MCKPQ: (1) Each pair of nodes in H' is adjacent or connected by at least one node in Q ; (2) $|H'| \geq |Q| + c = c'$; (3) for any node $v \in H, \deg_{G[H']}(v) = \deg_{G[H]}(v) + |Q| \geq |H| - k + |Q| = |H'| - k'$, for any node $v \in Q, \deg_{G[H']}(v) = |Q| - 1 + |H| = |H'| - k'$. So $G[H']$ is a k -plex. For the other direction, if we have a connected k -plex $G'[H']$ with size c' , then $G[H' \setminus Q] = G'[H' \setminus Q]$ is a k -plex by Lemma 1 and has size $c' - |Q| = c$, which completes the proof. \square

Theorem 3 *For any $\epsilon > 0$, it is hard to approximate for MCKPQ problem in polynomial time within a factor $n^{1-\epsilon}$.*

Proof Works in [16] show it is $n^{1-\epsilon}$ -hard to approximate the *maximum clique problem*, which aims to find the largest clique in a given graph. Given an instance G of MCP, we can perform a gap-preserving reduction to MCKPQ by setting $G' = G, Q = \emptyset$, and $k = 1$, whose solution is identical to MCP. So there is no $(n^{1-\epsilon})$ -approximation algorithm which runs in polynomial time for MCKPQ, where $n = |V(G)|$. \square

2.4 Problem Analysis

In this subsection, we first show that the solution of MCKPQ problem can avoid the free rider effect, then we present the properties of our proposed community model, and show that it can meet the requirements for community search problem, as summarized in Sect. 1.

2.4.1 Free Rider Effect(FRE)

In community detection problem, the *free rider effect* is under some goodness metric, the results of community

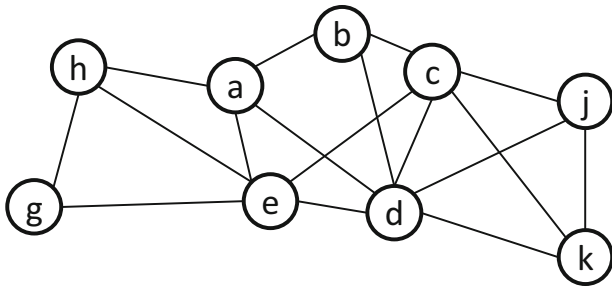


Fig. 1 An example graph

search admit including irrelevant subgraph [18, 30]. According to [18], most of community metrics suffer from free rider effect, such as minimum degree, graph density, subgraph modularity and so on. We first give an example of FRE and formal definition of it based on [18], and then we show the formulation of MCKPQ can avoid free rider effect.

Minimum degree $\delta(H)$ is commonly used as community measurement in [4, 11, 22, 28], the bigger, the better. Take graph G in Fig. 1 for example, suppose $Q = \{g\}$. Then most subgraphs in G consisting of $\{g, e, h\}$ could be as results of this query with optimal value 2, for example the subgraph $\delta(\{g, h, e, d, k, j\}) = 2$. But obviously it cannot be returned as best community, since k, j are too far away from g and make no contribution to optimal value. The same scenario also exists in k-truss model in [17].

Definition 6 (Free Rider Effect [18]) Suppose H is a solution to a community definition according to a goodness function $f(\cdot)$. The community definition is called suffers from free rider effect, provided whenever there is an optimal solution H^* to the community detection problem, then $f(H \cup H^*) \leq f(H)$. This means the combination of communities H and H^* is no worse than H .

Theorem 4 The definition of MCKPQ could avoid free rider effect.

Proof Given Q and k , the optimal community H^* is the largest k-plex containing Q . So for any other k-plex $\{H : Q \subset H\}$. And $f(\cdot)$ is simply $|H|$. There are two cases between H and H^* :

1. $H \subset H^*$. Then $f(H \cup H^*) = f(H^*) = |H^*| > |H| = f(H)$.
2. $H \not\subset H^*$. Since H^* is optimal and maximal, then $H \cup H^*$ cannot be k-plex (otherwise, $H^* = H \cup H^*$ since $H \cup H^*$ is larger) and becomes not feasible any more. We cannot evaluate $H \cup H^*$ by $f(\cdot)$.

In both cases, we cannot get $f(H \cup H^*) \leq f(H)$. So definition of MCKPQ could avoid FRE. \square

2.4.2 Properties of k-plex

In this section, we dive into properties of k-plex and describe how it can meet requirements of community search problem stated previously. These properties include bounded size, bounded diameter, connectivity.

Given single k , then any k-plex community has global bounded size.

Theorem 5 Given a connected graph $G(V, E)$, for any node v , its optimal k-plex size satisfies

$$|H^*| \leq \frac{(k + 2) + \sqrt{(k + 2)^2 - 8(|V| - |E|)}}{2} \tag{1}$$

Proof Suppose a k-plex H . Then $\delta(H) \geq |H| - k$ and $|E(H)| \geq \frac{|H|(|H| - k)}{2}$. Then, we contract H to one node in G to get G' . From the fact G is connected, G' is also connected. So $|E(G')| \geq |V(G)| - |H| + 1 - 1 = |V| - |H|$. Also $|E|$ is the upper bound of number of edges of this two graphs H and G' .

$$|E| \geq |E(H)| + |E(G')| \geq \frac{|H|(|H| - k)}{2} + |V| - |H| \tag{2}$$

Solving above inequation, we get the result. \square

If we take Q into consideration, then we have a local bounded community size stated as follows:

Theorem 6 Given an instance of MCKPQ(G, Q, k), the optimal size

$$|H^*| \leq \min_{q \in Q} (|N_G(q)|) + k \tag{3}$$

Proof Suppose H is one of solutions. Then $\delta(H) \geq |H| - k$, it follows $|N_H(q)| \geq |H| - k$ for any $q \in Q$. Since $H \subset G$, then $|N_G(q)| \geq |N_H(q)|$. So we have $|H| \leq k + |N_G(q)|$. Because arbitrariness of q , so $|H| \leq \min_{q \in Q} (|N_G(q)|) + k$. \square

We next discuss the connectivity of k-plex. We first show that for a disconnected k-plex, upper bound of its size is related to the number of its disjoint component and k . Based on that, we show when a k-plex is larger than a threshold, it must be connected.

Theorem 7 If disconnected k-plex $H = \bigcup_{i=1}^{\alpha} C_i$ has α disjoint components ($\alpha > 1$), then $|H| < \frac{\alpha}{\alpha - 1} (k - 1)$.

Proof Suppose C_i is the minimized component of H , then $|C_i| \leq \frac{|H|}{\alpha}$. For each node in C_i , it has at least $(\alpha - 1)$ non-neighbors, so $\alpha - 1 \leq k - 1 \Rightarrow \alpha \leq k$.

$$\begin{aligned}
 |H_2| &\leq |C_i| + (\alpha - 1) \\
 \Rightarrow |H_2| &\leq \frac{|H_2|}{\alpha} + (k - 1) \\
 \Rightarrow |H_2| &\leq \frac{\alpha}{\alpha - 1}(k - 1)
 \end{aligned}
 \tag{4}$$

□

Actually this bound is tight. Consider the disconnected 3-plex $V(H) = \{g, h, j, k\}$ in Fig. 1 with two components. $|H| = 2 * (3 - 1) = 4$.

Corollary 1 *If a k-plex H whose size is larger than (k + 1), then H must be connected.*

Proof We prove by contradiction. Suppose H is disconnected, then it has $1 < \alpha \leq k$ components, and $H \leq \frac{\alpha}{\alpha - 1}(k - 1)$. That implies $k + 1 < \frac{\alpha}{\alpha - 1}(k - 1)$ by Theorem 7. However,

$$\begin{aligned}
 k + 1 - \frac{\alpha}{\alpha - 1}(k - 1) &= \frac{(\alpha - 1)(k + 1) - \alpha(k - 1)}{\alpha - 1} \\
 &= 1 - \frac{\alpha - k}{\alpha - 1} > 0 \quad (\text{since } 1 < \alpha \leq k)
 \end{aligned}
 \tag{5}$$

the Eq. 5 contradicts with our assumption. □

Theorem 8 *If G is a connected k-plex, then $|G| \geq k + 1$.*

Proof Since G is connected, then every node in G would at least have one neighbor, which implies $|G| - k \geq 1$. □

From above we can see that when H is larger than (k + 1), it must be connected. As a result, while finding k-plex community, we do not need to do the connectivity checking for most of them, which is a main process in most community search algorithm [28, 31]. If we make the assumption for each k-plex community query, there always exists H such that $|H| > k + 1$, in this case, connectivity is not a concern any more.

Since diameter is another additional constraint for community query [18, 28]. Next, we show that any k-plex has bounded diameter by k.

Theorem 9 *If a connected k-plex G has the diameter d, then $k \geq d$.*

Proof Suppose $v_1 - v_2 \cdots v_{d+1}$ is the longest shortest path in G. For node v_1 , there are at least $d - 1$ non-neighbors in this path, which implies $k - 1 \geq d - 1$. □

Corollary 2 *All nodes of optimal solution of MCKP are in k-hop neighbors of Q.*

In this section, we state rationality of k-plex community search problem. Even though the problem is simple to model, interestingly it has nice properties for size,

connectivity, diameter, free rider effect, all of which are major requirements for most of community search problem. Next, we discuss solutions to MCKPQ problem.

First, we give two basic enumeration algorithms as baselines. Next, we improve by presenting branch-and-bound based algorithm and introduce the upper bound function and pruning strategies. Furthermore, we present two methods to accelerate the basic B&B algorithm.

3 Baseline Methods

We first give the baseline method to solve the MCKP problem. Since MCKP is NP-hard and it is hard to approximate by any linear function in polynomial time, we use the *generate and verify* method to explore the whole search space. Algorithm 1 describes this basic framework: It enumerates all k-plexes consisting of Q, the methods of enumeration differ but all using neighbors of current k-plex being enumerated, keep those with largest size (note that there maybe multiple largest k-plexes with the same size), among them it returns the connected one. Otherwise, it claims there does not exist solution for the query.

Algorithm 1 Basic Framework

Input: G, k, Q
Output: optimal k-plex community H
 1: $\mathcal{H} \leftarrow Enum(G, k, Q)$
 2: $H \leftarrow \arg \max_{H \in \mathcal{H}} |H|$
 3: **if** $\exists H^* \in H, H^*$ is connected **then**
 4: **return** H^*
 5: **else**
 6: there does not exist connected k-plex of Q

The search space of enumeration decreases with increasing query size, based on Theorem 10.

Theorem 10 *Given $Q = \{q_1, \dots, q_n\}$ and $Q_i = \{q_1, \dots, q_i\} (1 \leq i < n)$. Let M_i denote set of all maximal k-plex of Q_i . Then for any $1 \leq i \leq j \leq n, M_j \subset M_i$.*

Next, we show that if maximized k-plexes are not connected, then all k-plexes are not connected.

Theorem 11 *If all maximized k-plex of Q is not connected, then there does not exist connected k-plex consisting of Q.*

Proof We prove by contradiction. Suppose $H = \bigcap_{i=1}^n C_i$ is maximized k-plex of Q, disconnected. And there exists another connected k-plex H' that satisfies $|H'| < |H|$. Since the enumeration is based on neighbors of Q and H is disconnected, then $G[Q]$ is disconnected either, at the same time, each component C_i contains at least one of the query nodes. H' includes all query nodes, and H' would intersect each of component of H. Now H' can be represented by

$H' = (\bigcup_{i=1}^z I_i) \cup (H' \setminus H)$. We now can construct a connected k-plex H^* whose size is equal to $|H|$ by following steps. Initially set $H^* = H$, at each iteration remove the node with minimum degree among all the components and add one node from H' until all nodes in $H' \setminus H$ are added to H' , H^* is a connected k-plex, contradicting with all H are disconnected. \square

Algorithm 2 NaiveEnum

Input: G, k, H
Output: optimal k-plex community H
 1: $Cand \leftarrow \{v : v \in N_G(H), \delta(H \cup \{v\}) \geq |H| + 1 - k\}$
 2: **if** $Cand \neq \emptyset$ **then**
 3: **for all** $v \in Cand$ **do**
 4: $NaiveEnum(G, k, H \cup \{v\})$
 5: **else**
 6: **return** H

To achieve both maximization and connectivity constraint of MCKP, based on Algorithm 1, we only need to check the connectivity of those k-plexes with largest size with Theorem 11. The *NaiveEnum* (shown in algorithm 2) is used to generate all k-plexes of Q . Starting with Q , it searches each of Q 's neighbors to check whether it is validate to extend Q until there is no such neighbors. Even though *NaiveEnum* would generate all maximal k-plexes, it can result in repeated generation, i.e., enumerate identical k-plex multiple times.

Example 1 When $Q = \{g\}$ and $k = 2$ in Fig. 1, then the maximal 2-plex $\{g, a, e, h\}$ could be enumerated by $g \rightarrow e \rightarrow a \rightarrow h$ and $g \rightarrow h \rightarrow a \rightarrow e$. Hence, tedious computation may happen in naive enumeration.

To prevent stated drawbacks and reduce the computation, we design algorithm basing on *Bron-Kerbosch* algorithm [8], which is used to generate all maximal cliques recursively given a graph. Here, we changed it to enumerate all maximal k-plex containing a set of specific query nodes. At each iteration, three sets R, P, X are feed to *Maximal Search (MS)*, shown in algorithm 3. R is current founded k-plex, which is to be extended; P is candidate nodes, each of which can enlarge R , and nodes in X can also extend R but they are used in previous search. When there is no candidate can be used to enlarge R , then it is maximal. And if there are all candidates have already been used in previous search, i.e., X is not empty, it means that maximal k-plex containing R is already enumerated. Otherwise, we perform DFS search for all candidates. To get maximal k-plex of Q , MS is initialized with $R = \{Q\}, P = \{v : v \in N(Q), R = \{v : v \in N(Q), \{v\} \cup Q \text{ is k-plex}\}, X = \emptyset$.

Algorithm 3 B&KEnum

Input: R, P, X
Output: all maximal k-plex containing R
 1: **if** both P and X are empty **then**
 2: **yield** R ; $\{R \text{ is a maximal k-plex}\}$
 3: **if** P is empty **and** X is not empty **then**
 4: R can not be extended;
 5: **return**
 6: **while** $P \neq \emptyset$ **do**
 7: $v \leftarrow P.pop()$
 8: $R' \leftarrow R \cup \{v\}$
 9: $P' \leftarrow \{u | u \in N(R'), R' \cup \{u\} \text{ is a k-plex}\}$
 10: $X' \leftarrow X \cap P'$
 11: $P' \leftarrow P' \setminus X'$
 12: $MaximalSearch(R', P', X')$
 13: $X \leftarrow X \cup \{v\}$

4 Branch-and-Bound

The general *generate and test* procedure shown above is costly. First, for each found k-plex, it expands all its neighbors no matter whether they can lead to a better solution or not. Second, all candidates are enumerated equally, and each neighbor is chosen with equal probability to expand current solution. However, some candidates and neighbors are more potential to enlarge the size of current k-plex. To reduce the search space and improve efficiency, we develop *branch-and-bound(B&B)* based algorithm shown in Algorithm 4.

B&B paradigm is widely used for solving large-scale NP-hard combinatorial problems. An explicit enumeration for hard problem is normally time consuming due to exponentially increasing number of potential solutions. Using bounds for the function to be optimized plus score of current best solution enables B&B to search parts of the candidate space only.

Algorithm 4 is to get the maximized k-plex of Q . It is DFS-based branch-and-bound, which is shown efficient practical in solving various hard problems.

At each iteration, H is the current $k - plex$, we select validate nodes B (candidate neighbors are generated by *cand_gen*) from H 's neighbors and branch on each node of set B . Algorithm 4 also generates maximal k-plexes of Q ; however, it only searches partial searching space instead of all of them, which is performed in *NaiveEnum* and *B&KEnum*. This is done by following strategies:

1. define effective upper bound function for candidate, if current candidate k-plex cannot improve the quality better than current optimal one, then it is discarded. So the search space starting with this branch is removed.
2. for each candidate to be extended, select neighbors of candidate with high priority first to get optimal k-plex early.
3. prune invalidate neighbors.

4.1 Upper Bound Function

Upper bound function is crucial in B&B algorithm, loose upper bound would have no ability to prune search space. To derive efficient upper bound function of current k-plex H , we consider two cases.

Definition 7 (*tight nodes*) A node $v \in H$ is called *tight* if $deg_H(v) = |H| - k$

The upper bound function of H is defined as follows:

1. There exist tight nodes in H . Suppose $V = \{v : v \in H, v \text{ is tight}\}$. Then at most $|\bigcap_{v \in V} N(v)|$ nodes could be added to H , this means every node added to H must be one of neighbors of tight node v (any node non-neighborhood of v is unqualified because v has already $k - 1$ non-neighborhood in H).
2. There exist no tight nodes in H . In this case, upper bound is defined as $|H| + \min_{v \in H} (|N_{G \setminus H}(v)| + (k + N_H(v) - |H|))$. For each node, it can expand $|N(v)|$ neighbors and limited number of non-neighbors candidates, and the maximized size H can be expanded depends on lowest candidates number of $v \in H$.

Upper bound of above two cases involves $|N(v)|$. We further improve the bound quality by replacing $N(v)$ with $\{u : u \in N(v), \delta(H \cup \{u\}) \geq |H| + 1 - k\}$. By not counting neighbors that cannot be used to expand H , the gap between limitation of H and bound function is reduced.

Algorithm 4 Branch&Bound

Input: Q, G, k
Output: optimal k-plex community H

```

1:  $Cand \leftarrow \emptyset$ 
2:  $Cand.push((Q, \emptyset))$ 
3: while  $Cand \neq \emptyset$  do
4:    $H, Block \leftarrow Cand.pop()$ 
5:    $B \leftarrow cand\_gen(G, k, H)$  {generate all valid neighbors of  $H$ }
6:   while  $B \neq \emptyset$  do
7:      $v \leftarrow B.pop()$ 
8:      $H' \leftarrow \{v\} \cup H$ 
9:      $Block \leftarrow Block \cup \{v\}$ 
10:    if  $upper\_bound(H) < len(optimal)$  then
11:      continue
12:    else
13:      if  $|H'| > len(optimal)$  then
14:         $optimal \leftarrow H'$ 
15:         $Cand.push((H', Block))$ 

```

Algorithm 5 Basic Candidate Generation

Input: H, G, k
Output: candidate neighbors B

```

1:  $B \leftarrow \emptyset$ 
2: for all  $v \in H$  do
3:   for all  $u \in N(v), u \notin H$  do
4:     if  $H \cup \{v\}$  is k-plex then
5:        $B.add(u)$ 
6: return  $B$ 

```

4.2 Prune Unqualified Nodes

Unqualified nodes refer to those that are not able to produce maximized k-plex of Q or even k-plex of Q . Next, we introduce two strategies for pruning unqualified nodes: one is based on the query distance, the other is based on the core number.

Definition 8 (*Query Distance*) Given a graph $G(V, E)$ and a set of query nodes $Q \subset V$, and an arbitrary node $v \in V$, the query distance between v and Q is $dist(v, Q) = \min_{v \in Q} dist(v, q)$, where $dist(v, q)$ is the length of shortest path between v and q .

Based on Theorem 9, the query distance of any node v and Q is no greater than k . So the candidate nodes set $V_c = \bigcap_{i=1}^{|Q|} \{v : v \text{ is } k - \text{hop neighbor of } q\}$. We only perform the search in the subgraph $G' = G[V_c]$.

Furthermore, not all neighbors of current candidate H are supposed to branch on. Let current maximized k-plex is denoted by H^* . Neighbors of H are pruned by following strategies: *degree* pruning and *core number* pruning. Degree pruning is straightforward: if for node $v \in N(H)$, $deg_{G'}(v) < |H^*| - k$, then it is unqualified for branching. Since for any subgraph consisting of v , its minimum degree is no greater than $deg(v)$.

Definition 9 (*core number*) The *core number* c_v of node v is the highest order of a core that contains this node.

Theorem 12 For any subgraph $H \subset G$, if $v \in H$, then $\delta(H) \leq c_v$.

Proof This can be proved by contradiction. Let $\delta(H) > c_v$ and H' denote $\delta(H) - \text{core}$ of G , then $H \subset H'$, otherwise, $H' \cup H$ would result in a larger subgraph whose minimum degree is no less than $\delta(H)$, contradicting H' is maximized. Since $H \in H'$ and $v \in H$, then $c_v \geq \delta(H') = \delta(H)$. This contradicts that $\delta(H) > c_v$. \square

By Theorem 12, a node v cannot be included in any subgraph whose minimum degree is larger than c_v . So we remove nodes in $N(H)$ if $c_v < |H^*| - k$, where H^* is current best solution. Note that calculating core number for each node is a preprocessing step, and this can be done in linear time by core decomposition using method in [5].

We now analyze time complexity of basic candidate generation. Testing whether H is k-plex can be done in $O(|H|)$ time. The algorithm 5 would run in $O(d_{\max}|H|^2)$, where $d_{\max} = \max_{v \in H} deg_G(v)$.

5 Optimization on B&B

5.1 Partial Branching

In basic branch-and-bound algorithm, given a k-plex H , we simply extend H by branching all the validate neighbors B of H until all of them become maximal or cannot produce larger results by upper bound function. That is, we enumerate all branches and give them the same priority. The following two observations lead us to devise more subtle branching schema. First, the minimum degree of optimal k-plex is largest among others. Second, bottleneck of extending a current k-plex H always depends on the nodes those have the minimum degree.

So the improved candidate generation method *partial branching* makes following improvement:

- Instead of representing each current k-plex H as a set of nodes, it uses min-heap structure with node v as key, $deg_H(v)$ as value.
- At each iteration, we only consider neighbors of node with minimum degree in H and use them to expand H .

Next we show the partial branching would not miss any feasible solution.

Theorem 13 *Given any two k-plex H_0 and H_k in G , which satisfies $H \subset H'$, H' is connected. There always exists a sequence v_1, \dots, v_k such that $\forall i \in [0, k - 1], v_i \in H_k \setminus H_0, H_{i+1} = H_i \cup \{v_i+1\}, (v_{i+1}, u_i) \in E(G)$, where $u_i = \arg \min_{u_i \in H_i, \exists v \in H_k \setminus H_i, (u_i, v) \in E(u_i, v)} deg_{G[H_i]}(u_i)$.*

Proof We can construct this sequence by following steps. First, we partition $G[H_k]$ into two parts $A = G[H_0], B = G[H_k \setminus H_0]$. Then at each step, we move one node from B to A , by choosing the crossing edge(one endpoint in A , the other in B) whose degree of the node in A is minimum compared with other crossing edges. The process is always success until B is empty. Suppose at some step A', B' there is no crossing edge and B' is not empty, this would imply $G[A' \cup B'] = G[H_k]$ is not connected, which contradicts the assumption. \square

We now analyze complexity of partial candidate generation. Since we only extend neighbors of node v with minimum degree in H and H is a min-heap, fetching v can be done in $O(1)$ time. Let d_{\min} denote degree of node v . And up to d_{\min} , updates are performed to H due to expanding. Updating a heap H can be done in $O(\log(|H|))$. Totally, the time complexity is $O(d_{\min}|H|)$

5.2 Incremental Candidate Update

We now introduce an alternative to reduce cost of candidate neighbors generation. In basic B&B algorithm, in each round validate neighbors are generated from scratch given H . That would incur the case one node would be generated multiple times in following steps, starting from H .

Example 2 Consider the graph in Fig. 1 again, suppose $H = \{a, \}$, $k = 2$. Figure 2 shows one of search paths of basic B&B algorithm, extending H until it becomes maximal. At each round, it first gets its neighbors and keeps those valid. From B_1 (candidates in round 1) to B_5 , node c is generated and tested three times and node d twice. The case also applies to nodes not valid. For example, g is rejected in $N_3(H)$, but still tested in the next round in $N_4(H)$.

Algorithm 6 Incremental Candidate Update

```

Input:  $H, G$ , candidates in last round  $B_t$ 
Output: candidates in this round  $B_{t+1}$ 
1:  $H \leftarrow H \cup \{v\}; v = B_t.pop()$ 
2:  $can\_expand \leftarrow true$ ;
3: for all  $u \in B_t$  do
4:   if  $(v, u) \notin E(B_t)$  then
5:      $B_t[u] \leftarrow B_t[u] + 1$ 
6:     if  $B_t[u] \geq k$  then
7:        $can\_expand \leftarrow false$ 
8:     if  $B_t[u] > k$  then
9:        $B.remove(u)$ 
10: for all  $w \in H$  do
11:   if  $deg_H(w) = |H| - k$  then
12:      $can\_expand \leftarrow false$ 
13:   for all  $u \in B_t$  do
14:     if  $(w, u) \notin E(G)$  then
15:        $B_t.remove(u)$ 
16: if  $|H| > k$  then
17:    $can\_extend \leftarrow false$ 
18: if  $can\_extend = true$  then
19:   for all  $x \in N_{G \setminus (B_t \cup H)}(v)$  do
20:      $B_t.add\_key(x)$ 
21:      $B_t[x] \leftarrow |H|$ 
22:  $B_{t+1} \leftarrow B_t$ 
23: return  $B_{t+1}$ 
    
```

The above example shows tedious calculation in candidate generation step. Here, we propose more efficient way (shown in algorithm 6) to generate candidates B_{t+1} , which takes use of results B_t in last round. We define B as hash table, whose key is node id and value number of non-neighbors in H plus 1. First, remove a v of B_t and add it to

Fig. 2 Extending $H = \{a\}$ to maximal 2-plex

	H	Neighbors of H	Candidate Neighbors B
Round 1	{a}	{b,d,e,h}	{b,d,e,h}
Round 2	{a, b}	{c,d,e,h}	{c,d,e,h}
Round 3	{a,b,e}	{c,d,h,g}	{c,d}
Round 4	{a,b,d,e}	{c,j,k,h,g}	{c}
Round 5	{a,b,d,e,c}	{h,g,j,k}	{}

$H.can_expand$ indicates whether new members can be added. Then, we increase the value of node who is not in neighbors of v in B_t by one. Then filter out those non-neighbors in H exceed k . Next, we add new members x from $N_{G \setminus (B_t \cup H)}(v)$ if can_expand is still positive. There are three cases that we cannot extend the current candidates B_t from residual graph $G \setminus (B_t \cup H)$:

1. Some node u of B_t becomes tight. In this case, it would block all new members since any new node x . Because node $u \in B_t$, u must have at least one neighbor in H , which implies $|\overline{N_H(x)}| = |H| > |H| - 1 \geq |\overline{N_H(u)}| > = k$.
2. Some node w of H becomes tight. If $x \in N_{G \setminus (B_t \cup H)}(v)$, adding x would cause $|\overline{N_{H \cup \{x\}}(w)}| > |\overline{N_H(w)}| = k$.
3. $|H| > k$. Then, any new member would have at least k non-neighbors in H .

In above three cases, can_expand would be toggled to negative and $|B_t|$ would decrease in the following iterations.

We now analyze time complexity of candidates update. Updating values of hash table costs $O(|B_t|)$ time. Checking tight nodes in H costs $O(|H||B_t|)$ time. Adding new member cost $O(\ln(x))$ time if B_t can be extended. Since $|B_t|$ always smaller than $N(H)$, the total time complexity would be $O(|H||N(H)|)$.

6 Heuristic Solutions

Since MCKPQ is hard for both optimization and approximation, in this section, we propose heuristic algorithms which are fast and produce high-quality results, while have no theoretical guarantee. At each step, we choose one node from $N_G(H)$ greedily to extend H until H becomes maximal. The new node is chosen based on the quality function f . Intuitively, given a set of neighbor nodes to extend H , the larger $f(v)$, the more promising $H \cup \{v\}$ could reach a k-plex with large size.

So how to define a quality function is the key issue to influence results. Given two k-plex H and H' with same

size, there are two aspects to compare its ability of extension:

1. Number of missing links. K-plex requires each node in it has at most k non-neighbors, so H would be more promising to expand to larger size if $|E_G(H)| > |E_G(H')|$.
2. neighbors of H . This is straightforward because if $N(H) > N(H')$, H would have a higher chance to become a larger community.

So we define the quality function of v as follows:

$$f_H(v) = |E_G(H \cup \{v\})| + \frac{|N(H \cup \{v\})|}{\max_{v \in C} |N(H \cup \{v\})|} \tag{6}$$

The intuition is to find neighbors which can achieve largest edges increment and break ties by choosing the one who can add more neighbors to current solution.

Algorithm 7 Heuristic

```

Input:  $Q, k, G$ 
Output:  $H$ 
1:  $H \leftarrow Q$ 
2: while  $H$  is k-plex do
3:    $C \leftarrow \{v : v \in N_G(H), H \cup \{v\} \text{ is k-plex}\}$ 
4:    $v \leftarrow \arg \max_{v \in C} f_H(v)$ 
5:    $H \leftarrow H \cup \{v\}$ 
6: return  $H$ 

```

7 Experiment

We have conducted extensive experiments to test the effectiveness and efficiency of our proposed solutions in finding k-plex community. We first show by case study that the effectiveness of k-plex community model capturing cohesive membership structure and common interests of members in searching results. Then, we report the experimental results on the efficiency tests using different problem settings on different datasets. We use four real-world datasets provided by Stanford network dataset collection:⁵ (1) DBLP collaboration network; (2) Amazon product co-

⁵ <https://snap.stanford.edu/>.

Table 2 Statistical information of datasets

Dataset	#Nodes	#Edges	Avg. deg	diam(<i>G</i>)
DBLP	317,080	1,049,866	6.62	21
Amazon	334,863	925,872	5.53	44
Arxiv COND-MAT	23,133	93,497	8.08	14
Google-Web	875,713	5,105,039	9.87	21

purchasing network; (3) Google-Web graph; (4) Arxiv collaboration network of condensed matter. The statistical information of datasets is shown in Table 2. All algorithms are implemented in Python and run on a PC with Intel(R) Core(TM) i7 CPU 860 @2.8GHz 2.8GHz with 8GB memory.

7.1 Case Study

Since different community query models differ in their objective goodness functions (such as degree, density, and diameter), an uniform quantitative quality evaluation is

beyond this paper’s scope. Instead, we show the results of our k-plex community search by case studies.

We use the dblp coauthor graph, in this graph two scientists are linked together if they have worked on the same paper. We issue the query $Q = \{Jiawei Han\}$, $k = 2$ and we get optimal communities shown in Fig. 3, both of which achieve maximized community. To see the effectiveness, we can see members in H_1 are all data mining scientists. Jiawei Han and Philip S. Yu have worked on about 50 papers together, and Jian Pei and Ke Wang have worked on more than 20 papers together. H_1 and H_2 also indicate the ability of extracting overlapping communities of k-plex model. By increasing k by 1 and using the same query, we get another result shown in Fig. 4a, which is a supergraph of H_1 , this implies the hierarchical structure of k-plex community model. We also issue query with multiple nodes $Q = \{Xuemin Lin, Jeffrey Xu Yu\}$, $k = 3$. From the result shown in Fig. 4b, we can see members in this community all have published papers in graph database and processing, which are one of research interests of Xuemin Lin and Jeffrey Xu Yu and this is verified by checking their webpage profile.

Fig. 3 Communities of $Q = \{Jiawei Han\}$, $k = 2$. **a** 2-plex community H_1 and **b** 2-plex community H_2

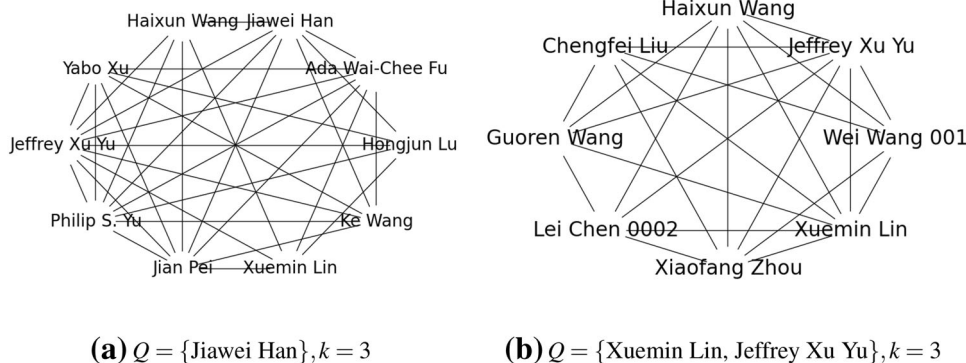
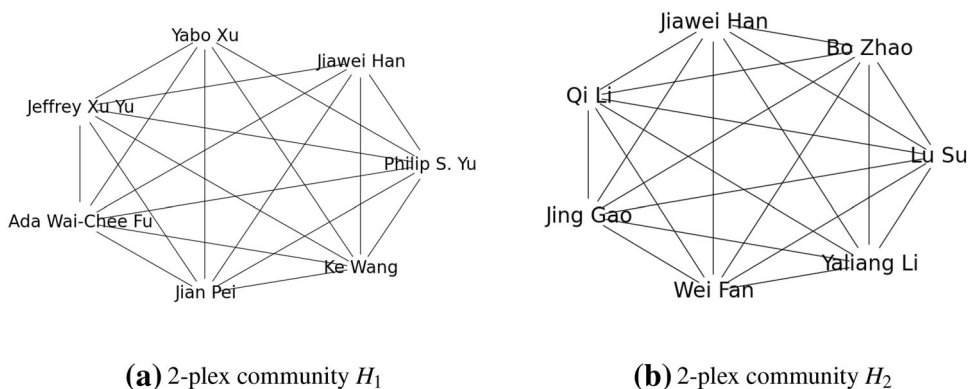


Fig. 4 Results of 3-plex community query. **a** $Q = \{Jiawei Han\}$, $k = 3$ and **b** $Q = \{Xuemin Lin, Jeffrey Xu Yu\}$, $k = 3$

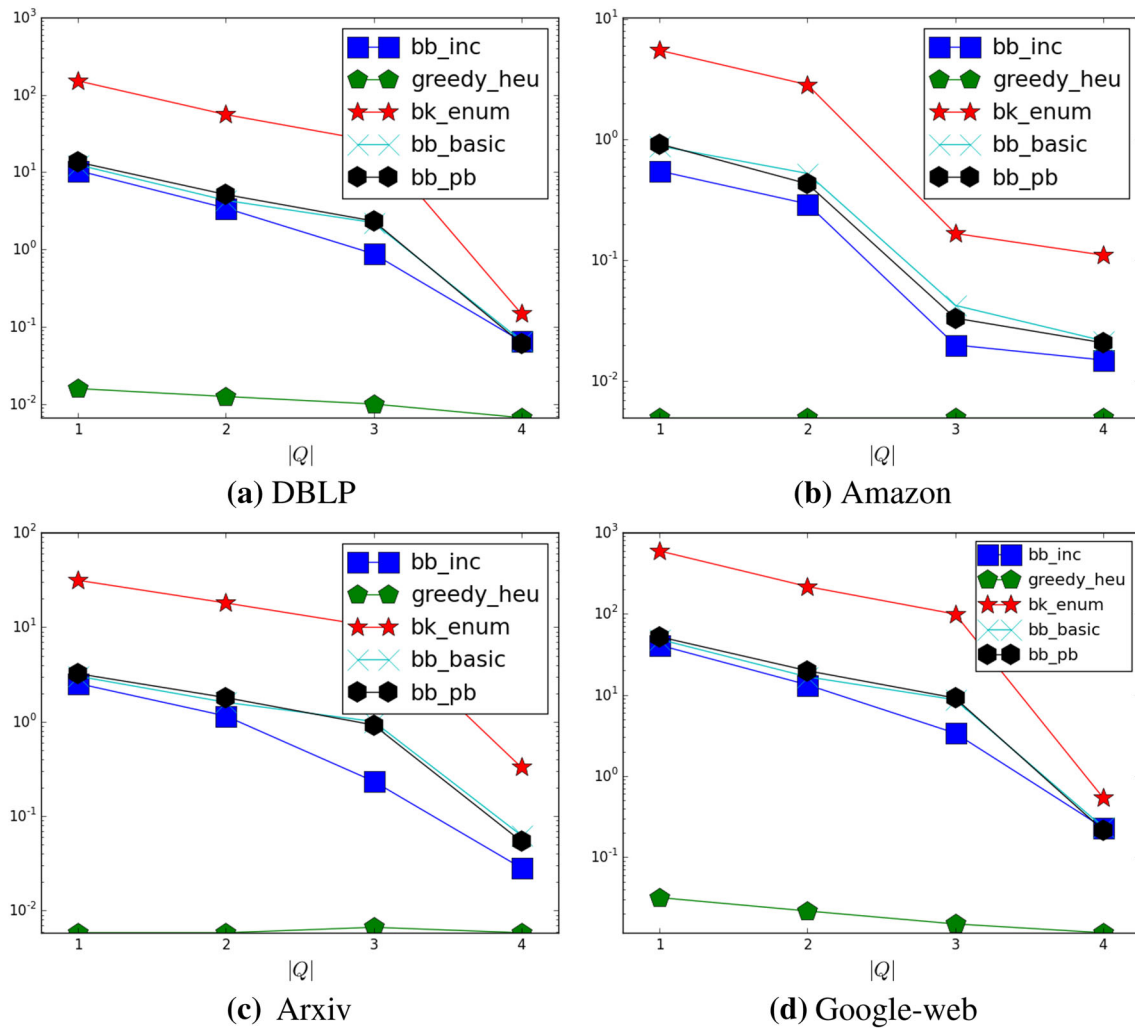


Fig. 5 Query time varying $|Q|$ (aggregated on $k \in [1, 4]$). **a** DBLP, **b** Amazon, **c** Arxiv and **d** Google-Web

7.2 Efficiency Results

Next, we study the time efficiency of different methods. There are two parameters for querying k -plex community: Q and k . We range $|Q|$ from 1 to 4 and k from 1 to 4. To generate query sets, some preprocessing steps are conducted, for example: For each graph, we first sort all the nodes based on their degree. Then, nodes are partitioned into five buckets in degree order so that each bucket has the same number of nodes. For each pair of $(|Q|, k)$, we randomly sample 20 queries from each degree bucket. So for each query setting, we get 100 queries to evaluate. Diameter of query is also concerned, let Q_k denote query nodes for k , then queries are generated with the restriction of $diam_G(Q_k) \leq k$, since for any $diam_G(Q_k) > k$, there does not exist feasible solution. We denote B&K enumeration by **bk_enum**, the naive branch-and-bound method by **bb_basic**, the partial branching method by **bb_pb**, the

incremental candidate generation method by **bb_inc**. Since most of results of the naive enumeration method do not return in hours thus cannot be collected, we would ignore this method in our comparison.

7.2.1 Varying $|Q|$

We evaluate querying time using different query size $|Q|$. For each $|Q|$, we aggregate query time on $|Q|$ and calculate the average. The results of different datasets are listed in Fig. 5. For all datasets, B&K enumeration always has highest query delay. And branch-and-bound based methods can reduce query delay significantly. The simple greedy heuristic solution has lowest cost of time. For different B&B based methods, **bb_inc** outperforms **bb_basic** and **bb_pb** since its incremental candidate generation reduces the repeated evaluation of neighbors. The improvement of **bb_pb** on **bb_basic** is not notable. The reason is that even

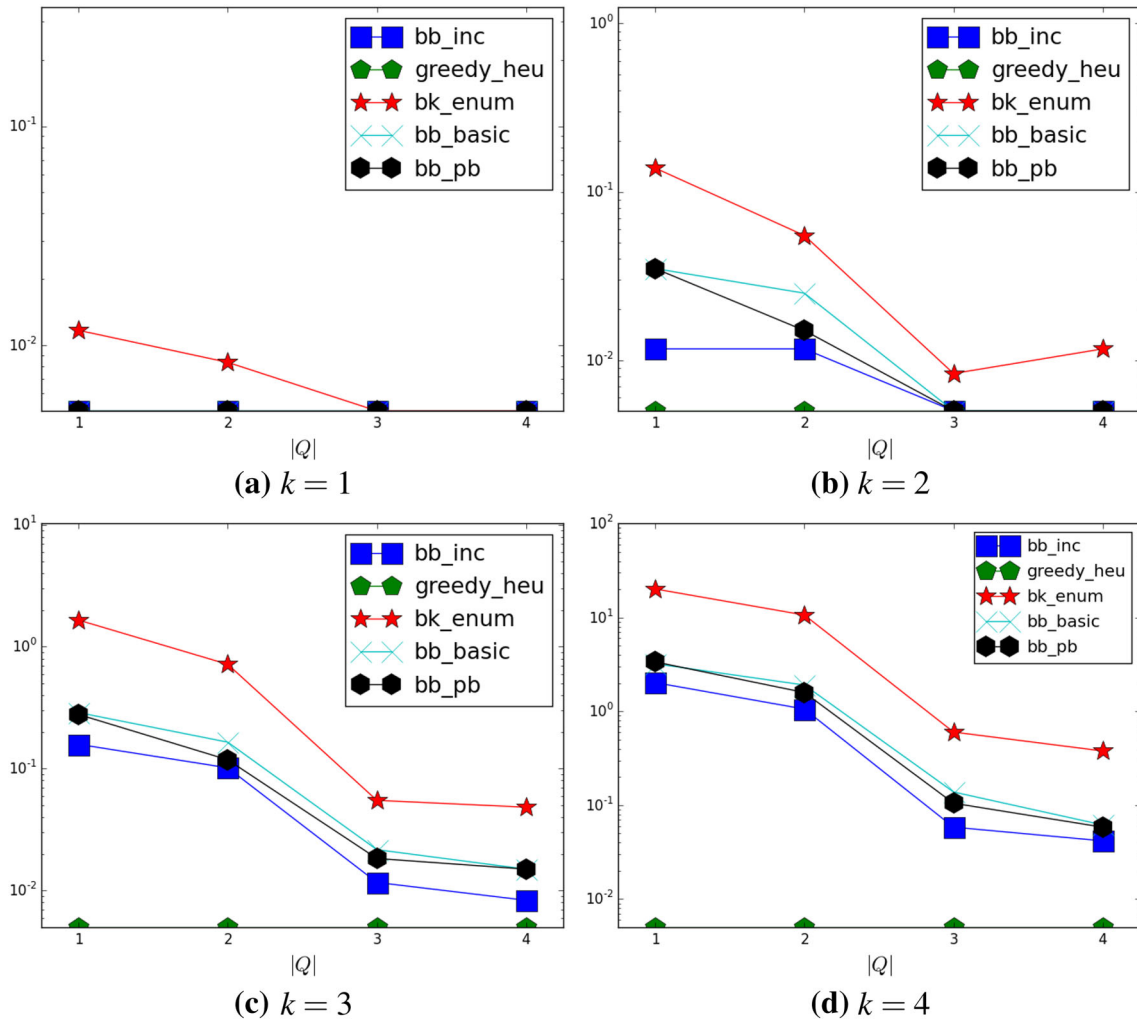


Fig. 6 Amazon: query time varying $|Q|$. **a** $k = 1$, **b** $k = 2$, **c** $k = 3$, and **d** $k = 4$

though partial branching decreases the cost of selecting valid neighbors, i.e., it narrows down the number of children of a branching node in each step, the size of search tree is not shrunk. We can also observe that query time decreases with increasing $|Q|$. This is due to search space is reduced by adding query nodes. This is illustrated by Theorem 10. While $|Q| = 1$, searching for maximum k-plex would cost hundreds of seconds. When $|Q|$ has increased to 4, query time decreases, respectively, to between 0.01 and 0.1 s. Average query time of each $|Q|$ in Fig. 5 shows the aggregated results on various k . Thus, to get a more clear investigation of varying $|Q|$, we also fix k for various query sizes. Results of Amazon dataset are shown in Fig. 6, and evaluation of other datasets is omitted due to limited space. For $k = 1$, the task is equal to search for maximum clique and the B&K enumeration is most costly compared with other methods, while they could get near zero cost for clique finding.

7.2.2 Varying k

We test query performance on various k . Even though there is no limitation of k in k-plex search, usually k is set to small values to keep subgraph cohesive, 2 and 3 are frequently used in practice [3, 6]. We range k from 1 to 4. For each specific k , different query size $|Q|$ is evaluated and we take the average query time. Results from different datasets are shown in Fig. 7. The query time increases with increasing k . This is because the larger k , the larger upper bound of number of non-neighbors; this results in larger search space. B&K enumeration is most costly in different k settings. And B&B based algorithm is better, due to its pruning technique and early termination. In spite of improvement of branch-and-bound technique, the complexity is still exponential with k , since it is an exact solution. And the heuristic greedy algorithm always terminates fast and outperforms others. For fixed k , results of different query size contribute to average query time. Since queries with small $|Q|$ contribute large amount of query

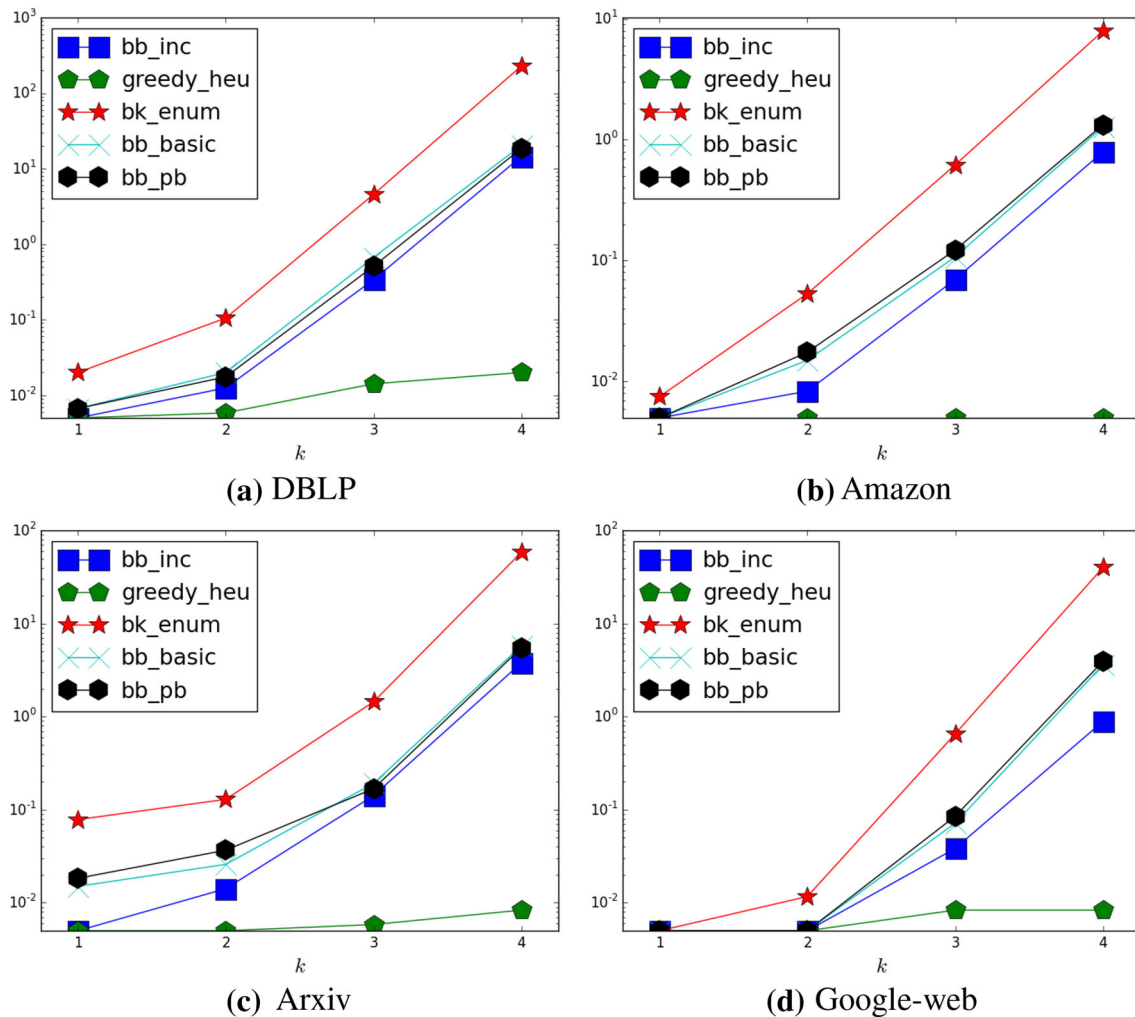


Fig. 7 Query time varying k . **a** DBLP, **b** Amazon, **c** Arxiv, and **d** Google-Web

time. We also test results of different fixed $|Q|$ to get a more accurate evaluation of various k , shown in Fig. 8. Results of DBLP dataset are listed and others omitted. As we can see from Fig. 7, while $k = 1$, query time is no larger than 0.1 s. When k is increased to 4, the time consuming would increase to hundreds of seconds. This is due to the exponential time complexity of MCKPQ on k .

7.2.3 Quality of Heuristic

The quality of output of heuristic greedy algorithm is also tested, shown in Fig. 9. Since all other methods would output optimal k -plex community, so only one of exact algorithm bb_basic is compared. For each specific k , average size of optimal k -plex is calculated. We feed the same set of queries to heuristic solution and get average size of output. We can see from results heuristic solution has size larger than 50% of optimal averagely. The results also imply the influence of k on the size of output

community. In average, smaller k would result in smaller cohesive community.

8 Related Work

8.1 Densest Subgraph Problem

Densest subgraph problem is a major research topic in graph analysis. Given a graph G , the task is to find the densest subgraph. Average degree is one of most frequent used measurements in dense subgraph mining. [15] first gives polynomial time algorithm $O(mn)$ for densest subgraph problem, by transforming it into min-cut instance and using binary search to get the optimal density. An 2-approximation algorithm is proposed in [9], this is done by greedily removing node which is of minimum degree. This greedy algorithm runs in $O(m + n)$ time. [19] first introduces densest subgraph problem for directed graphs and

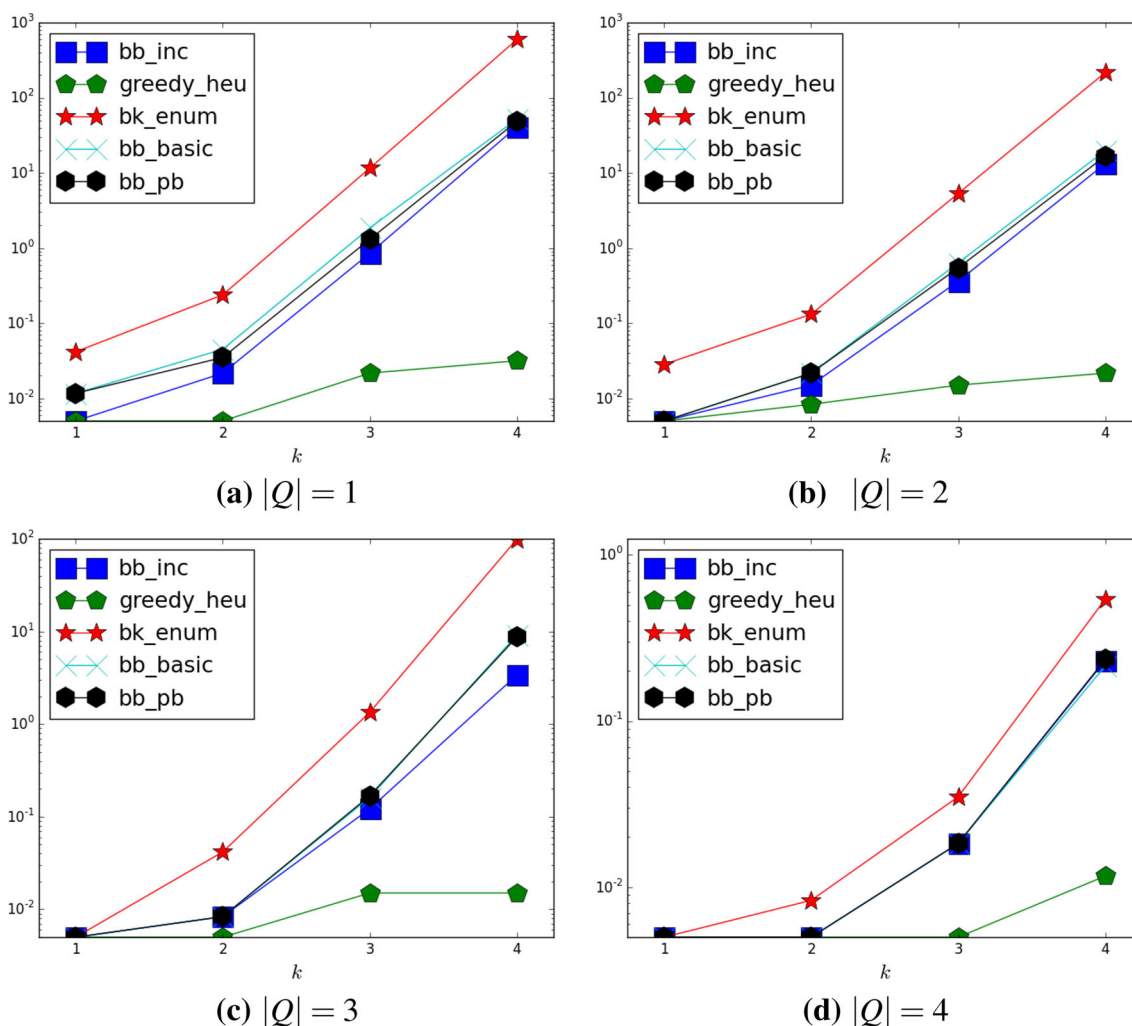


Fig. 8 DBLP: query time varying k . **a** $|Q| = 1$, **b** $|Q| = 2$, **c** $|Q| = 3$ and **d** $|Q| = 4$

gives an $O(\log(n))$ -approximation algorithm. [20] then gives a max-flow based exact algorithm firstly and improves the greedy one to $O(m + n)$.

Even though densest subgraph problem (DSP) admits polynomial time complexity, it becomes NP-hard when there is a size constraint. There are three variants: (a) k -densest subgraph problem (DkS): It requires subgraph $|S| = k$; (b) densest subgraph at least k (DalkS): It requires subgraph $|S| \geq k$; and (c) densest subgraph at most k (DamkS): It requires subgraph $|S| \leq k$. It is hard to approximate DkS and DamkS problem within a constant factor. [12] introduces an $O(n^{\frac{1}{2}})$ -approximate algorithm for DkS. However, DalkS can be approximated with a constant factor. An 3-approximate solution is shown in [2]. Furthermore, [20] provides an 2-approximate algorithm for Dalks.

Our work differs in that it's query specific and its goodness metric is size of k -plex, instead of subgraph density.

8.2 Community Search

Community search problem is recently proposed in [28]. The task is to find high-quality community given initial query members. There is no uniform goodness metric for this problem. Minimum degree measurement is used in [4, 11, 28]. K -truss based variants are introduced in [17, 18]. [17] requires results are of *edge connectivity* to avoid disjoint communities. They further define optimal k -truss community by largest k and smallest diameter in [18]. Since the problem becomes NP-hard, they present $(2 - \epsilon)$ -approximate algorithm to solve it. [30] addresses the free rider effect and proposes query-density based community query problem. [29] studies triangle densest community problem and shows a max-flow based algorithm. However, the connectivity of answer community is not guaranteed. [10] studies searching overlapping communities given one single query node. Their model is based on quasi-clique.

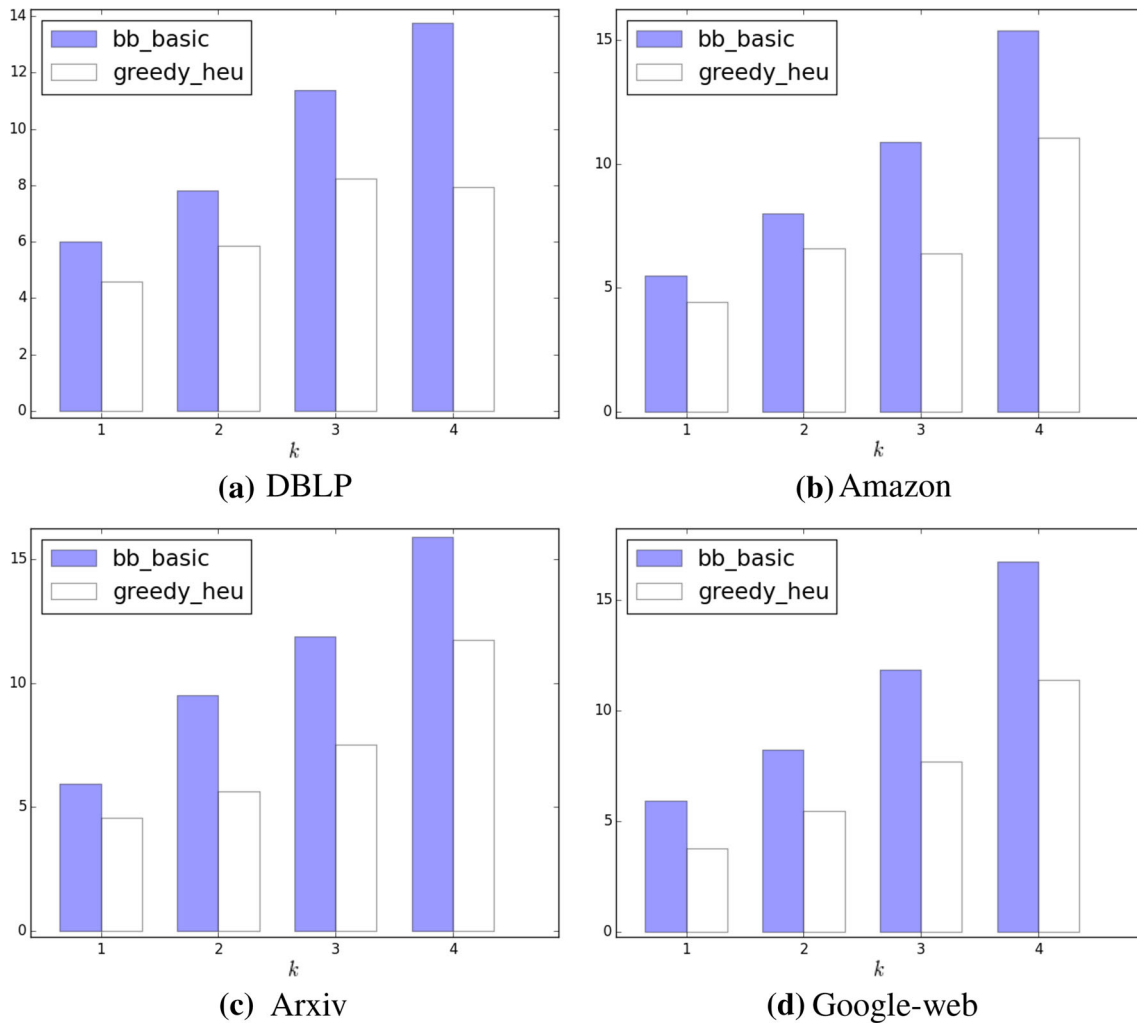


Fig. 9 Quality comparison of heuristic greedy. **a** DBLP, **b** Amazon, **c** Arxiv and **d** Google-Web

Our work differs from above works in definition of objective community and unrestricted size of query nodes.

8.3 Community Detection

Community detection is a well-studied topic in graphs and social network analysis. The task is to identify and list all communities given a graph. A common used measurement is modularity [24]. Modularity maximization is shown NP-hard theoretically in [7]. [1] introduces rounding technique and efficient algorithms. However, optimizing modularity in large networks has limitation to resolve small communities, shown in [14]. The above works focus on detecting disjoint communities and graphs are partitioned. Since each person can be involved in multiple groups in social network, there are also works on detecting all overlapping communities [32]. Our work differs in that it is query oriented. And query result can be different from that of

maximizing modularity (a global optimal graph partition may not be an local optimal community for query sets).

9 Conclusion

In this paper, we study querying optimal k-plex community problem, that is, given a set of nodes Q , finding optimal k-plex community consisting of Q . We show that our proposed community model can guarantee the good quality of query results theoretically. Based on the fact the problem is NP-hard and hard to approximate, we design efficient branch-and-bound method and further improve it by technique of fast generating candidates. We then give heuristic solution of low time cost. Experiments show the effectiveness of our k-plex model and efficiency of our proposed methods.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Agarwal G, Kempe D (2008) Modularity-maximizing graph communities via mathematical programming. *Eur Phys J B* 66(3):409–418
- Andersen R, Chellapilla K (2009) Finding dense subgraphs with size bounds. In: Proceedings of algorithms and models for the web-graph, 6th international workshop, WAW 2009, Barcelona, Spain, February 12–13, 2009, pp 25–37
- Balasundaram B, Butenko S, Hicks IV (2011) Clique relaxations in social network analysis: the maximum k-plex problem. *Oper Res* 59(1):133–142
- Barbieri N, Bonchi F, Galimberti E, Gullo F (2015) Efficient and effective community search. *Data Min Knowl Discov* 29(5):1406–1433
- Batagelj V, Zaversnik M (2003) An $O(m)$ algorithm for cores decomposition of networks. arXiv preprint cs/0310049
- Berlowitz D, Cohen S, Kimelfeld B (2015) Efficient enumeration of maximal k-plexes. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data, Melbourne, Victoria, Australia, May 31–June 4, 2015, pp 431–444
- Brandes U, Delling D, Gaertler M, Gorke R, Hofer M, Nikoloski Z, Wagner D (2008) On modularity clustering. *IEEE Trans Knowl Data Eng* 20(2):172–188
- Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. *Commun ACM* 16(9):575–577
- Charikar M (2000) Greedy approximation algorithms for finding dense components in a graph. In: Proceedings of approximation algorithms for combinatorial optimization, third international workshop, APPROX 2000, Saarbrücken, Germany, September 5–8, 2000, pp 84–95
- Cui W, Xiao Y, Wang H, Lu Y, Wang W (2013) Online search of overlapping communities. In: Proceedings of the 2013 ACM SIGMOD international conference on management of data, ACM, pp 277–288
- Cui W, Xiao Y, Wang H, Wang W (2014) Local search of communities in large graphs. In: International conference on management of data, SIGMOD 2014, Snowbird, UT, USA, June 22–27, 2014, pp 991–1002
- Feige U, Kortsarz G, Peleg D (2001) The dense k-subgraph problem. *Algorithmica* 29(3):410–421
- Fortunato S (2010) Community detection in graphs. *Phys Rep* 486(3):75–174
- Fortunato S, Barthélemy M (2007) Resolution limit in community detection. *Proc Nat Acad Sci* 104(1):36–41
- Goldberg AV (1984) Finding a maximum density subgraph. University of California, Berkeley, Berkeley
- Håstad J (1999) Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math* 182(1):105–142
- Huang X, Cheng H, Qin L, Tian W, Yu JX (2014) Querying k-truss community in large and dynamic graphs. In: International conference on management of data, SIGMOD 2014, Snowbird, UT, USA, June 22–27, 2014, pp 1311–1322
- Huang X, Lakshmanan LVS, Yu JX, Cheng H (2015) Approximate closest community search in networks. *PVLDB* 9(4):276–287
- Kannan R, Vinay V (1999) Analyzing the structure of large graphs. Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn
- Khuller S, Saha B (2009) On finding dense subgraphs. In: Proceedings of automata, languages and programming, 36th international colloquium, ICALP 2009, Rhodes, Greece, July 5–12, 2009, Part I, pp 597–608
- Li K, Lu W, Bhagat S, Lakshmanan LVS, Yu C (2014) On social event organization. In: The 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '14, New York, NY, USA, August 24–27, 2014, pp 1206–1215
- Li R, Qin L, Yu JX, Mao R (2015) Influential community search in large networks. *PVLDB* 8(5):509–520
- McDermott R (2000) Knowing in community. *IHRIM* 19
- Newman ME (2004) Fast algorithm for detecting community structure in networks. *Phys Rev E* 69(6):066,133
- Rowe L, Nadeau J, Turner R, Frankel W, Letts V, Eppig J, Ko M, Thurston S, Birkenmeier E (1994) Maps from two interspecific backcross dna panels available as a community genetic mapping resource. *Mamm Genome* 5(5):253–274
- Seidman SB, Foster BL (1978) A graph-theoretic generalization of the clique concept. *J Math Sociol* 6(1):139–154
- Song DM, Tumminello M, Zhou WX, Mantegna RN (2011) Evolution of worldwide stock markets, correlation structure, and correlation-based graphs. *Phys Rev E* 84(2):026,108
- Sozio M, Gionis A (2010) The community-search problem and how to plan a successful cocktail party. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 939–948
- Tsourakakis C (2015) The k-clique densest subgraph problem. In: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp 1122–1132
- Wu Y, Jin R, Li J, Zhang X (2015a) Robust local community detection: on free rider effect and its elimination. *Proc VLDB Endow* 8(7):798–809
- Wu Y, Jin R, Li J, Zhang X (2015b) Robust local community detection: on free rider effect and its elimination. *PVLDB* 8(7):798–809
- Xie J, Kelley S, Szymanski BK (2013) Overlapping community detection in networks: the state-of-the-art and comparative study. *ACM Comput Surv (csur)* 45(4):43