World Wide Web (2022) 25:569–607
https://doi.org/10.1007/s11280-021-00891-6

Finding attribute diversified community over large attributed networks

Afzal Azeem Chowdhary[1] · Chengfei Liu[1] · Lu Chen[1] · Rui Zhou[1] · Yun Yang[1]

Accepted: 28 April 2021 / Published online: 17 July 2021

Abstract

Well connected users are generally discovered in communities which is one of the most important tasks for network data analytics and has tremendous real applications. In recent years, community search in attributed graphs has begun to attract attention, which aims to find communities that are both structure and attribute cohesive. Meanwhile, searching a community that is structure cohesive but attribute diversified, denoted as attribute diversified community search, is still at an early stage. In this paper, we introduce our recent effort for discovering attribute diversified community. In fact, for different applications, the needs of attribute diversification for modelling the community are quite different. We introduce three attribute diversified community models in which attribute diversification takes different roles for presenting as an objective and as a constraint. We also discuss major techniques for speeding up the attribute diversified community search. We conduct extensive experiments to show the effectiveness and efficiency of our algorithms for finding attribute diversified communities in various settings.

Keywords Community search · Diversity · User engagement · Attributed networks · Indexing

This article belongs to the Topical Collection: *Special Issue on Large Scale Graph Data Analytics*
Guest Editors: Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang

✉ Chengfei Liu
cliu@swin.edu.au

Afzal Azeem Chowdhary
achowdhary@swin.edu.au

Lu Chen
luchen@swin.edu.au

Rui Zhou
rzhou@swin.edu.au

Yun Yang
yyang@swin.edu.au

[1] Swinburne University of Technology, Melbourne, Australia

1 Introduction

Graphs have emerged as a powerful model for representing different types of data, such as *social networks, author collaboration networks, location based social networks (LBSNs), citation networks, etc.* In these graphs, discovering communities that naturally exist as groups of fine-connected users is one the most important tasks for network data analytics and has tremendous real applications. The networks properties are necessary to make sense of communities that are formed by a set of entities that are densely connected with certain relationships. Density of subgraph using structure cohesiveness is one of the state of the art methods to detect communities. The nodes in these networks will represent entities such as users, authors, influencers, etc. While the edges represent friendships, co-authorship, follower relationship, etc. Several distinct models for cohesive (dense) subgraphs and communities have been proposed in the literature that are based on structure cohesiveness, e.g., k-core [4, 39], k-truss [15], k-edge-connected graph [8, 9, 53], k-vertex-connected graph [44], etc.

Nevertheless, most of the previous studies [4, 8, 9, 39, 44, 53] have focused on finding communities from a graph without considering attributes. As such, the returned communities may miss out important attributes describing a variety of features of real applications. Recently, *attributed community search* has been studied well in literature and has gathered some attention. Normally, the entities in a real-life networks often exhibit properties which are beneficial to make sense of communities through graphs. Such networks are coined as *Attributed Graphs*. Here, the attributes of the nodes help search the affluent communities which are both structure and attribute cohesive. These state of the art methods use structure cohesiveness alongside attribute cohesiveness to search attributed communities, e.g. (k, d)-truss [1], (k, r)-core [50], (k, d)-MCC [11], k-truss CAC model [55]. These works endeavour to find communities that are both structure and attribute cohesive. Since attribute cohesiveness is an optional constraint to search communities, communities are also found where attributes are diversified [12, 31, 34]. Here, the attributes of the nodes exhibit certain level of diversity within the community. However, study for community search that takes serious consideration of structure cohesiveness but attribute diversification within a community is still at an early stage.

In this paper, we study about *attribute diversified community*. By an attribute diversified community, we mean that the connected nodes inside the community exhibit difference in terms of interested attributes. In addition, different purposes for diversification may end up with different kinds of attribute diversified communities. Therefore, we introduce and propose attribute diversified community search, including three attribute diversified community models in which attribute diversification takes different roles for being presented, as an objective, and as a constraint. The first two community models proposed consider attribute diversification as an objective. And, the last one considers attribute diversification as a constraint.

1.1 Maximising attribute diversification

Discovering a community with members as diversified as possible has numerous applications. The relationships between connected members exhibit high diversity, i.e., diverse in terms of attributes shared but also being cohesive with regard to their friendships/social relationships. Here, we present attribute diversity as a search objective that needs to be maximised. We propose two different search objectives, the max-min problem and the max-average problem.

An example for the max-min problem, is finding a set of users from Twitter with various technological expertise to form a diverse collective representation. Such users are generally representative members of the group that maintain high attribute diversity between the relationships, where members are technically sound, and the relationships can capture various technological expertise. The high attribute diversity is maintained when the worst possible attribute diversity value of the relationships is the best minimum value that can exist across the community.

An example for the max-average problem, is building a team for group brainstorming to address a cognitive bottleneck of idea generation. Group brainstorming shall engage diversified individuals to collaborate by communicating and sharing ideas in groups, where diversified individuals can substantially broaden the knowledge base available for idea generation and the social engagements among the individuals allow the creative effort to be aggregated. The attribute diversity of the relationships is aggregated to maximise the overall diversity of the community. For such applications, since they target community members for innovations and there are evidences that maximising the overall diversity leads to creativity [41], the desired community would be preferred to maximise the overall attribute diversity of its members, which was previously studied in our work [14]. This max-average problem differs from the above max-min problem, which considers the worst possible attribute value if they can contribute to the overall average diversity of the community.

1.2 Attribute diversification as constraint

Some applications would like to find a community that exhibits certain level of attribute diversification but has members with social relationships as cohesive as possible. One such example is in Facebook, where we find a cohesive user set with distinct linguistics abilities for promoting products across multi-lingual pages/groups for better reachability. The cohesiveness of the group will make the communication and promotion easy. Other applications include, identifying connected influencers with diverse topic orientation in social contagion modelling, finding sequences exhibiting anomalies in gene/protein regulation, forming a socio-cultural contrasting group for providing expert recommendation, etc. For these applications, the target members show high degree of cohesion that leads to better communication between the users. We call such a community as the *attribute diversified community with maximum cohesiveness*. This allows the users to be greatly involved to get a trans-informative result and distinctive enough in terms of their user attributes. The preliminary idea of this problem was introduced in [34].

1.3 Contributions and road map

We assimilate the models that we have proposed above and list their contributions. Then, we present the road map that shows the flow of the paper. The main contributions of this paper are as follows:

– We aim to find an attribute diversified community where attribute diversification is considered as a search objective to be maximised by considering it as a max-min problem (Section 3).
– We then aim to find an attribute diversified community where attribute diversification is considered as a search objective to be maximised by considering it as a max-average problem (Section 4).

– We also aim to find the most cohesive attribute diversified community where attribute diversification is used as a constraint to find the most structurally cohesive diversified community (Section 5).

The rest of this paper is organised as follows. In Section 2, we introduce and discuss basics for attributed graphs. In Sections 3–5 we discuss our attribute diversified community search works. We conduct extensive experimental evaluations and present them in Section 6. We discuss the related works and conclude the paper in Sections 7 and 8 respectively.

2 Preliminaries

In this section, we formally introduce the commonly used community cohesiveness metrics.

An attributed graph is denoted as $G = (V, E, A)$, where $V(G)$, $E(G)$ and A denote the set of vertices G, the set of edges in G, and the set of attributes A in terms of *keywords* respectively. Each vertex $v \in V(G)$ is attached with a set of attributes $A_v \subseteq A$. Given $v \in V(G)$, $deg(v, G)$ denotes the degree of v in G and $N(v, G)$ denotes the neighbours of v in G.

Next, we define the social cohesiveness metrics used in paper.

Definition 1 ***k*-core subgraph.** Given a subgraph $H \subseteq G$, an integer k, H is called k-core subgraph if for every $v \in V(H)$, $deg(v, H) \geq k$ and such maximum k is called the coreness of H.

Definition 2 Coreness. Given a graph G, the coreness of a vertex $u \in V(G)$, denoted by c_u, is the largest k, such that u is contained in the k-core of G.

Intuitively, a k-core is a subgraph in which vertex has at least k neighbours. A k-core with a large value k indicates strong internal connections over members. A k-core is maximal if it cannot be extended.

Next, each H_k represents the set of all connected k-core subgraph in G and is called a ***k-core set***. For any integer $k' \geq k$, $H_{k'} \subseteq H_k$, i.e., the k'-core set is always a subgraph of the k-core set.

The ***k*-shell** S_k represents the set of vertices in G with *coreness* equal to k, i.e., $S_k(G) = V(H_k(G)) \setminus V(H_{k+1}(G))$.

Next, we briefly describe some properties of k-core, which we will use to define our indices further.

The idea behind hierarchy of k-cores is that, all the k-core cores of a graph are nested into one another, i.e., a $(k + 1)$-core is contained in a k-core. Thus, the following properties hold for every integer k. *a*) *Disjointness* For the same k-core set, every k-core is disjoint from each other in that set. *b*) *Containment* A k-core is contained by exactly one $(k-1)$-core. Therefore, the hierarchy of k-cores can be represented by a set of trees where each node is associated with a particular k-core set with coreness k. On this basis, we define the following.

Definition 3 ***k*-core tree node**. Given a k-core subgraph $H \subseteq G$, each k-core tree node T_H is uniquely associated with the vertices in H with coreness k, i.e., $H \cap S_k \neq \phi$.

Note a k-core tree node does not necessarily have vertices connected, because the k-shell vertices may be separated by other vertices with higher coreness.

A k-core tree node is a parent tree node if for any integers i, j and T_{H_i}, T_{H_j} are respectively associated with a k-core tree node, then T_{H_i} is the parent of T_{H_j} if,

(i) $k_i < k_j$; (ii) $H_j \subseteq H_i$; and (iii) any k'-core tree node with $k_i < k' < k_j$ is not the parent of T_{H_j}.

Given a k-core H is associated with a unique T_H, the k-core tree node contains vertices in the k-shell while leaving the rest in its children. The original H can be reconstructed recursively by the vertices in T_H and its children $T_{H_1}, \ldots T_{H_c}$, if T_H has children.

Definition 4 ***k*-core tree**. Given a graph G, the k-core tree is constituted by all k-core tree nodes for every integer k from 0 to k_{max}, where every core tree node is connected to its parent if it exists.

The above core tree organizes all k-cores into a hierarchy, where each tree corresponds to a connected component of the original graph. The core tree can be stored in $\mathcal{O}(n)$ space, because each vertex exists in exactly one core tree node corresponding to its coreness and each core tree node has exactly one parent.

While explaining our advanced search order heuristics, we use the following definitions as the basis to explain later in Section 5.2.

Definition 5 A graph G is k-degenerate if every subgraph g of G has a vertex with degree at most k in g. The **degeneracy** of G, denoted $\delta(G)$, is the smallest value of k for which G is k-degenerate. That is, the degeneracy $\delta(G)$ of G equals the maximum value among the minimum vertex degrees of all subgraphs of G.

Definition 6 **Degeneracy order.** Given a graph G, a permutation $(v_1, v_2, \ldots, v_n)$ of all vertices of G is a degeneracy ordering of G if every vertex v_i has the minimum degree in the subgraph of G induced by $(v_i, \ldots, v_n)$.

The k-degeneracy and the degeneracy ordering of the vertices of a given graph G can both be computed in linear time [39].

These are some common notations summarised in Table 1 used throughout the paper for simplicity.

We will introduce detailed attribute diversification metrics when introducing the specific models.

3 Attribute diversified community - max-min problem

In this section, we talk about maximising attribute diversity as a max-min problem. The motivating example will show how on using attribute diversity as a search objective can solve the max-min problem. We then introduce an attribute diversified community search work that maximises the max-min attribute diversity. Later on, we discuss the solutions.

Motivating example Let Figure 1 be a graph representing a Twitter network, where each node represents a user and each edge represents friendship between users. Each user in the graph has some distinct attributes which shows his/her domain areas/interests. Now, considering a scenario where to find a small representative community, the representation of distinct domain areas is maximised by a few number of members. Since, these members are knowledgeable in their respective fields and show distinction within the community.

Table 1 Notations used in the paper

Notation	Definition
G	an undirected attributed graph
H, H'	a subgraph of G
n, m	the total number of vertices and edges in G
$V(H), E(H)$	the vertices and edges in H
u, v	the vertices in the attributed graph
A_u	the attributes of u in G
(u, v)	an edge in the attributed graph
$deg(u, H)$	the degree of u in H
k_{max}	the max coreness of G
c_v	the coreness of vertex v of G
$\delta(G)$ or δ	the degeneracy of G
δ_v	the degeneracy of vertex v in G
S_k	the k-shell of G
H^*	the optimum result for an algorithm
h	a connected k-core

Therefore, we need a representative community whose relationships are attribute diversified. The *attribute diversified community model* that maximises attribute diversity by considering it as a max-min problem, is used to find a group of members that represents the diverse domains. The max-min problem finds the community by only keeping those relationships inside the community where the least possible attribute diversity value that can exist is the maximum one, while still being structurally cohesive. The max-min problem will try to leave out the worst values of attribute diversity from the community, and will only maximise the best of the minimum possible value. The advantage of such a representative community is that it can be found in a cost-effective manner. In Figure 1, each member has a relationship with at least 2 other members, which makes the graph structurally strong.

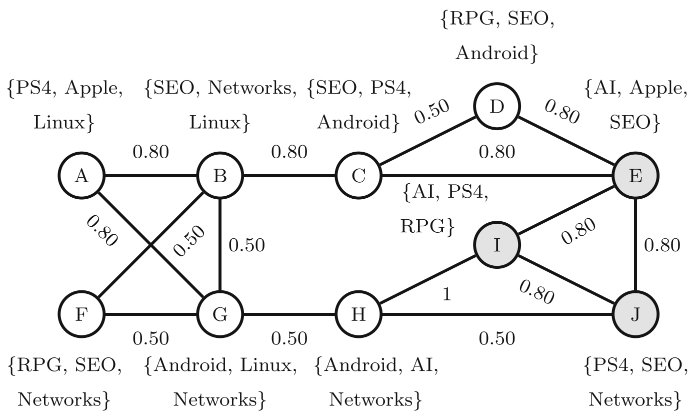


Figure 1 A twitter graph

In Figure 1, {E,I,J}, shown with shaded nodes, is the representative community found using the max-min problem. The max-min problem leaves out all the low values to attain the maximum possible least attribute diversity value of 0.8 across all the relationships, when each member knows at least 2 others in the community. Next, we study on how to model and search attribute diversified communities that maximises attribute diversity by considering it as a max-min problem. To model such an attribute diversified community, we first define a diversity measure on the relationships. The community's structure cohesiveness is ensured using k-core, where each member has a relationship with at least k users in the community. Based on the diversity measure and the k-core structure constraint, we model an attribute diversified community that maximises the max-min problem. We then develop efficient algorithms to solve the problem. We first show that, this problem can be solved in polynomial time by proposing a baseline algorithm, and then we propose an enhanced algorithm with less time complexity.

3.1 Problem definition

We first propose the diversity between any two vertices as follows.

Diversity for two vertices Given a pair of vertices $u, v \in G$ with attributes A_u and A_v respectively, the diversity function is defined as $div((u, v)) = 1 - \frac{|A_u \cap A_v|}{|A_u \cup A_v|}$.

The above diversity function defined between two vertices is based on the Jaccard distance metric. There are other distance metrics such as *cosine distance, edit distance, etc.*, but they are not suitable for defining the diversity function for our problem, which is to compute the similarity between two sets of attributes. We use Jaccard distance because it uses set operations, where the ordering of attributes is not important. But for cosine distance and edit distance which consider the similarity of two vectors or lists, the ordering of the attributes is important.

Next, we propose the minimum based diversification metric.

Minimum based diversity Given H, the minimum based attribute diversification of H is measured by

$$minDiv(H) = min\{div((u, v))|(u, v) \in E(H)\}.$$

Next, we consider member diversification as well as structure cohesiveness over members simultaneously to model a community. Therefore, we propose the member diversified community model, using k-core and minimum based diversity metric.

Attribute diversified community - max-min Now, we are ready to propose the attribute diversified community model based on max-min search objective.

Definition 7 Attribute diversified community - max-min. Given a subgraph $H \subseteq G$, an integer k, H is defined as an attribute diversified community if H satisfies the following constraints simultaneously:

- Connectivity: H is connected;
- Social cohesiveness: H is a k-core subgraph;
- Attribute diversity: H follows $minDiv(H)$.

Accordingly, given G and an integer k, the research problem we focus on in this paper is as follows.

Research Problem. ***Attribute diversified community search - max-min.*** *Find the subgraph $H \subseteq G$ that maximises $minDiv(H)$.*

Example To briefly show the results of the max-min problem. We will discuss the example shown in Figure 2a, where the edges are assigned random attribute diversity values on the edges. For the max-min problem, when k=3, the attribute diversified community is the subgraph {1,3,4,6} with maximum of minimum diversity 0.60 as shown in Figure 2c. Similarly, when k=2, the subgraph {5,6,7} with maximum of minimum diversity 0.77 is shown in Figure 2b.

In the following section, we firstly show that the max-min problem can be solved in polynomial time by proposing a baseline algorithm. Then we propose an enhanced algorithm with less time complexity, by taking the advantages of sort and incremental computation. This ensures that the problem can be solved in linear time.

3.2 Baseline approach

We first show that the decision version of the max-min problem can be solved in polynomial time. Then we show that we only need to solve at most m numbers of the decision problem. We start from the definition of decision version of max-min problem.

Decision version of max-min problem Given a graph G, an integer k and an edge (u, v) with attribute diversity of $div((u, v))$, we determine if there is subgraph $H \subseteq G$ such that H satisfies the first two constraints in Definition 3.1 while the $minDiv(H)$ is at least $div((u, v))$.

Lemma 1 *The decision version of max-min problem can be solved in $\mathcal{O}(m)$.*

Proof sketch We prove this lemma by showing an $\mathcal{O}(m)$ algorithm solving the decision problem. The algorithm is described as below. Firstly, we remove all edges with attribute diversity below $div((u, v))$, which takes $\mathcal{O}(m)$ time. Then, we progressively delete vertices with degree less than k from the remaining graph till all the remaining vertices with degree no less than k, which takes $\mathcal{O}(m)$. If the remaining graph is not an empty set then the result of decision problem is yes otherwise no.

Next, we show that the max-min problem can be solved in polynomial time by proposing a lemma below.

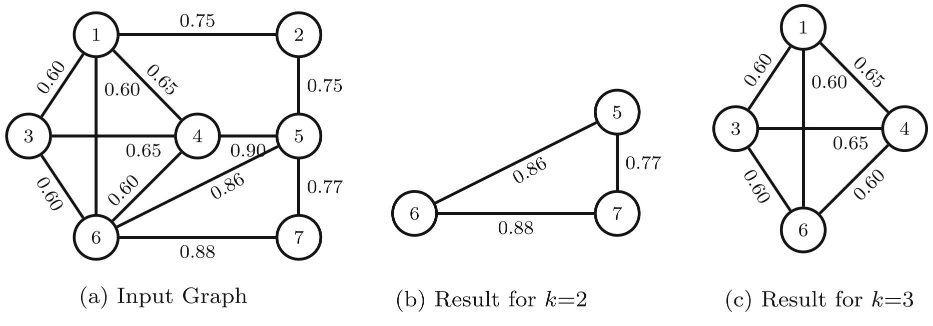


Figure 2 A toy example with attribute diversity for max-min problem

Lemma 2 *The* max-min *problem can be solved by solving at most m numbers of its decision version problem.*

The proof is trivial since there are at most m distinct values of $minDiv$ that we need to try for G before the optimum result can be derived. We are ready to show the complete baseline algorithm as follows.

Algorithm 1 Baseline.

```
   Input  : $H$: maximal $k$-core
   Output: $H^*$
1  $H^* \leftarrow H$;
2  for $(u, v) \in E(H)$ do
3      $H' \leftarrow H$, $H' \leftarrow$ remove edges in $H'$ with edge diversity less than $div((u, v))$;
4      $H' \leftarrow$ compute the maximal $k$-core for $H'$;
5      if $minDiv(H') > minDiv(H^*)$ then
6          $H^* \leftarrow H'$;
7      end
8  end
9  return $H^*$
```

Baseline algorithm Algorithm 1 shows the detailed steps. It takes a maximal k-core as the input since the result can only be contained in k-core subgraph if it exists. Next, it assumes the maximal k-core as the optimum results H^* (Line 1), solves all possible decision version of the max-min problems and improves H^* (Lines 2 to 8) during the iteration. After that, the algorithm returns optimum result H^* and terminates (Line 9).

The correctness of Algorithm 1 is clear based on Lemmas 1 and 2. The time complexity of Algorithm 1 is $\mathcal{O}(m^2)$ since the running time of the algorithm is dominated by Lines 2 to 8 that take $\mathcal{O}(m^2)$.

3.3 Enhanced approach

Algorithm 1 is inefficient for large graphs, since its time complexity is quadratic. In this section, we propose a faster online algorithm with time complexity of $\mathcal{O}(m \log m + m)$. With some precomputations, its time complexity can be reduced to $\mathcal{O}(m)$.

Intuition The key idea of the enhanced approach is as follows. If we can solve decision problems with d in non-increasing orders, where d is the edge diversity $div((u, v))$, the computations of lines 3 to 4 in Algorithm 1 can be done incrementally on progressively reduced subgraphs, which would make the time complexity of the loop become $\mathcal{O}(m)$, i.e., the time complexity of the loop in Algorithm 1 is reduced substantially.

Enhanced algorithm We list detailed steps of the enhanced approach in Algorithm 2. It takes maximal k-core as input and sorts edges in the k-core in non-increasing order based on their edge diversity. Next, it solves decision problems with progressively increased d while maintaining the current best result since edges are sorted (Lines 3 to 7). The decision problems are solved on a progressively reduced H as shown in Lines 4, 5 and Lines 8 to 16. Since all the edges are sorted, when the current decision problem has no result, the algorithm returns optimum result H^* and terminates.

Algorithm 2 Enhanced algorithm.

```
Input  : $H$ : $maximal\ k\text{-}core$
Output: $H^*$
1  sort edges in $E(H)$ in non-increasing order according to edge diversity;
2  $H^* \leftarrow H$;
3  for $(u, v) \in E(H)$ do
4      $E(H) \leftarrow E(H) \setminus \{(u, v)\}$;
5      $H \leftarrow$ incCore($H$);
6      If($H \neq \emptyset$) then $H^* \leftarrow H$ else return $H^*$ ;
7  end
8  Procedure incCore ($H$)
9      while $v \in \{V(H) | deg(v, H) < k\}$ do
10         for $u \in N(v, H)$ do
11             $E(H) \leftarrow E(H) \setminus \{(u, v)\}$;
12             $deg(u, H) \leftarrow deg(u, H) - 1$;
13         end
14         $V(H) \leftarrow V(H) \setminus \{v\}$;
15     end
16     return $(V(H), E(H))$;
```

Efficient implementation We discuss optimisations when implementing Algorithm 2.

Efficient sorting. When sorting the edges in a maximal k-core, we sort edges in each connected maximal k-core and then merge them. As such, the real running time is much better than $\mathcal{O}(mlogm)$.

Constant time edge deletion while preserving order. After sorting, we link edges in the maximal k-core and loop the edges using the linkage. As such, when deleting an edge, i.e., Lines 4 and 12, we can delete them from the graph in constant time while remaining edges preserve the sorted order. With those optimisations, Algorithm 2 runs in $\mathcal{O}(mlogm + m)$ time. If we consider sorting as precomputation, the time complexity of Algorithm 2 is $\mathcal{O}(m)$.

The experiments are reported in detail in Section 6.1.

4 Attribute diversified community - max-average problem

In this section, we talk about maximising attribute diversity as a max-average problem. The motivating example shows using the attribute diversity as a search objective to solve the max-average problem. The average attribute diversity will express the overall diversity of the community. We then introduce an attribute diversified community search work that satisfies the max-average attribute diversity. Afterwards, we discuss the solutions.

Motivating example Figure 3 shows an IMDb graph, where each node represents a person working in Hollywood and the relationship represents the work they have collaborated on. Each person's attributes describe the different genres they have worked in. Now, consider a scenario where to organize a film festival, we need a panel of acquainted jurors to judge a wide variety of films. The acquaintance of jurors ensures that the decision making process remains smooth because conflicts often exist implicitly in unestablished relations. In this scenario, we may have some relationships between users whose attribute diversity may be moderate. However, their differences with others still contribute significantly to the

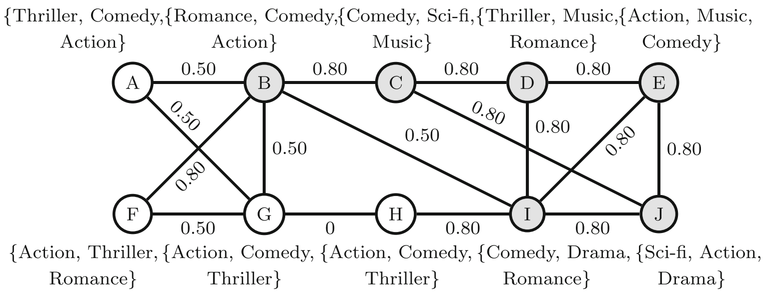


Figure 3 An IMDb graph

diversity of the community as a whole. Consequently, we need to find a community whose overall diversity is maximised along with it being structurally cohesive. The *attribute diversified community model* that maximises the average attribute diversity by considering it as a max-average problem, can be used to find a set of connected jurors whose overall attribute diversity express the vast expanse of the industry. From Figure 3, we see that nodes such as {A,F,G,H} are not quite diverse enough w.r.t. each other in terms of attributes. Here, each person is connected to at least 2 other persons which makes the graph structurally cohesive.

This problem is different from the max-min problem presented previously in Figure 1 in a way that the max-min problem can leave out these worst possible values of attribute diversity. Whereas, max-average problem can keep them because on leaving them out from the community we can lose these nodes causing the violation of the structure constraints, even when these nodes can contribute to the overall diversity of the community.

In Figure 3, the subgraph induced by {B,C,D,E,I,J} shown as shaded nodes, is the overall attribute diversified community found with the max-average problem, when each member knows at least 2 other members in the community. For max-average problem, the worst values of attribute diversity do not contribute much but without them the k-core constraint cannot be satisfied. Thus, it will disqualify this community as a candidate even though the max-average could be the best. For example, even though the attribute diversity between {B,I} is moderately the worst, they still contribute to the community to increase its overall attribute diversity. Whereas, the max-min problem will leave out such values of attribute diversity.

We now study on how to model and search attribute diversified communities that maximises the average attribute diversity by considering it as a max-average problem. To do this, the attribute diversity on the relationships is first defined through a diversity measure. The relationships will maximise the overall average attribute diversity of the community.

The structure cohesiveness of a community is guaranteed by using k-core, to ensure that each user has at least k neighbours in the community. Based on the diversity measure and the k-core structure cohesiveness constraint, we model an attribute diversified community that maximises the max-average problem. We first prove the NP-hardness of the problem, and then propose efficient branch and bound algorithms with novel effective bounds.

4.1 Problem definition

Here, we use the diversity between the two vertices to propose the average based diversification metric.

Average based diversity Given H, the average attribute diversification of H is measured by

$$avgDiv(H) = \frac{\sum_{(u,v)\in E(H)} div((u,v))}{|V(H)|}.$$

Next, we consider attribute diversification as well as structure cohesiveness to model a community. We propose the attribute diversified community model, using k-core and average based diversification metric.

Attribute diversified community - max-average Now, we are ready to propose the attribute diversified community model based on max-average search objective.

Definition 8 Attribute diversified community - max-average. Given a subgraph $H \subseteq G$, an integer k, H is defined as an attribute diversified community if H satisfies the following constraints simultaneously:

- Connectivity: H is connected;
- Structure cohesiveness: H is k-core subgraph;
- Attribute diversity: H follows $avgDiv(H)$.

Accordingly, given G and integer k, the research problem we focus on in this paper is as follows.

Research Problem. ***Attribute diversified community search - max-average***. Find the subgraph $H \subseteq G$ that maximises $avgDiv(H)$.

Example To briefly show the result of the max-average problem, we will discuss the example shown in Figure 4. Here, the edges are assigned random attribute diversified values. For the attribute diversified community - max-average problem with $k = 2$, the result is the subgraph {2,3,6,7,8,9,10,12,13} with diversity of 1.44 as shown in Figure 5a. Similarly, for $k = 3$, the result is the subgraph {3,8,9,10,13} with diversity of 1.418 as shown in Figure 5b.

Before showing efficient algorithms for solving the problem of finding attribute diversified community maximising average edge diversity, we first study the hardness of the problem.

Lemma 3 *The max-average problem is NP-hard.*

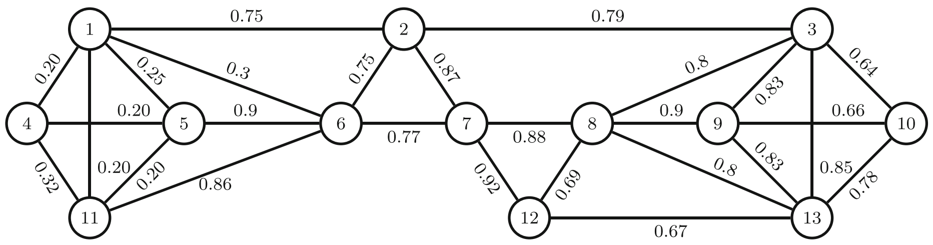


Figure 4 A toy example with attribute diversity

Proof sketch We reduce well known NP-hard problem, densest at least size ρ problem, denoted as $dals\rho$, to an instance of the max-average problem. Given a graph G, a size constraint ρ, $dals\rho$ problem finds the subgraph H with the highest density, defined as $\frac{|E(H)|}{|V(H)|}$, among all subgraphs that have size no less than ρ. Given any instance of $dals\rho$ problem with G and ρ, we construct an instance of max-average problem with k and G' as follows. k is set to be $\rho - 1$. G' is created based on G as follows. G' is a complete graph with $V(G') = V(G)$. The edge diversity of $e \in E(G')$ is assigned 1 if $e \in E(G)$, otherwise 0. With such construction, every subgraph in G' with size greater than k is a k-core. The answer for the max-average problem in G' if it exists can be used to derive the answer for $dals\rho$ by removing edges with edge diversity of 0. On the other hand, the answer for $dals\rho$ in G if it exists can be transferred to the answer for the created instance of the max-average problem by making the answer as a complete graph. It is easy to see that both the instance construction and the answers equivalence check can be done in polynomial time.

Due to the NP-hardness, there is no polynomial algorithm for solving the max-average problem. We will propose branch and bound algorithm for solving max-average problem, in which we will propose upper bounds and search order optimisations to reduce the search space as much as possible.

4.2 Search framework

For ease of understanding, we first show the basic enumeration used in the branch and bound algorithm.

Algorithm 3 shows the basic enumeration that derives the optimum result. Initially the input of the algorithm is G. By recursively calling itself, Algorithm 3 tries all possible subgraphs of G if the subgraphs may contain the optimum result (Lines 12 to 16) and checks if there is feasible solution in the current recursion (Lines 8 to 10). If there is a feasible solution h in the recursion and the feasible solution is greater than the current optimum one H^*, H^* is updated to h (Line 9).

Algorithm 3 basicADC.

```
   Input  : the input graph $G$, and integer $k$
   Output: the optimal solution $H^*$
1  $H^* \leftarrow \phi$;
2  basicEnum ($G$);
3  return $H^*$;
4  Procedure basicEnum ($H$)
5      $H' \leftarrow k$-core(H);
6      let $\mathcal{H}'$ be the set of connected component in $H'$;
7      foreach $h \in \mathcal{H}'$ do
8          if $avgDiv(h) > avgDiv(H^*)$ then
9              $H^* \leftarrow h$;
10         end
11     end
12     foreach $h \in \mathcal{H}'$ do
13         foreach $v \in V(h)$ do
14             basicEnum ($h \setminus \{v\}$);
15         end
16     end
```

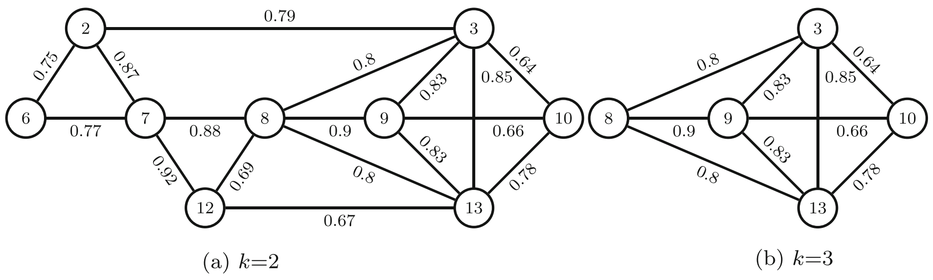


Figure 5 Result for max-average problem

Search space reduction Algorithm 3 also applies space reduction optimisations based on observations as follows.

Observation 1 *The optimum result can only be contained in a connected k-core of G if it exists when the enumeration starts.*

Observation 2 *During the recursion with an input H, the optimum result can only be contained in a connected k-core of H.*

With the observations, when a recursion starts, Algorithm 3 first reduces the input to the maximal k-core, which would make the input become a set of maximal connected k-core. Algorithm 3 only tries combinations in each connected k-core. As such, the search space can be reduced significantly.

The correctness of Algorithm 3 is clear. This is because all combinations of vertices that potentially lead to the optimum result are explored by the algorithm. However, it cannot scale to even medium size datasets. In the following sub-section, we will propose upper bounds and study upper bound based prunings to improve the search performance. To make the prunings more effective, we also propose heuristic rules to find subgraphs with large diversity as early as possible. Those optimisations improve the performance substantially.

4.3 Optimisation

Upper bound based pruning The idea is that we estimate the upper bound of the average edge diversity of the current search branch. If the upper bound is smaller than the diversity of the optimum result found up to the time, we terminate the search branch.

Next, we will propose three upper bounds.

Upper bound based on core property We firstly show an upper bound for a connected k-core based on core property. The upper bound for h is defined as follows.

$$ubcore(h) = \frac{\sum_{(u,v)\in E(h)} div((u, v))}{k+1} \tag{1}$$

The upper bound based on core property would only be tight when h contains an optimum result with size close to $k + 1$. However, it has limited pruning effectiveness when h contains large size results. Next we study tight bounds for arbitrary h.

Maximum average diversity in a core Given a connected k-core h, this bound is defined as follows.

$$ubavg(h) = max\{avgDiv(h')|h' \subseteq h\} \tag{2}$$

Lemma 4 *ubavg(h) is an upper bound for h.*

Proof Let h^* be the attribute diversified community contained in h and h' be $argmax_{h'}$ $\{avgDiv(h')|h' \subseteq h\}$. By definition since h^* must be a k-core while h' relaxes the k-core constraint. As such $h^* \subseteq h'$ must hold. Therefore, $avgDiv(h^*) \leq avgDiv(h')$ must hold. Otherwise, h' will not be $argmax_{h'}\{avgDiv(h')|h' \subseteq h\}$. We finish the proof. □

Approximate maximum average diversity in a core The computational cost of $ubavg(h)$ is expensive. It would take $\mathcal{O}(|V(h)|^3)$ if using the algorithm in [23]. However, there is a simple yet effective approximate algorithm [7] that can achieve $\frac{1}{2}$ approximation. As such we can use the approximation algorithm to get an at least $\frac{1}{2}$ $ubavg(h)$ value first and then multiple it by 2 to derive a slightly loose bound, denoted as $apxubavg(h)$. In implementation, $ubcore(h)$ and $apxubavg(h)$ are prioritised as they are cheap.

Search order For each connected k-cores that cannot be pruned, we sort them in non-increasing order according to their upper bounds. By doing this, we can heuristically find community with large average diversity as early as possible. This would make the upper bound based pruning more effective.

Algorithm 4 advADC.

```
   Input  : the input graph $G$, and integer $k$
   Output: the optimal solution $H^*$
1  $H^* \leftarrow \phi$;
2  advEnum ($G$);
3  return $H^*$;
4  Procedure advEnum ($H$)
5      $H' \leftarrow k$-core(H);
6      let $\mathcal{H}'$ be the set of connected component in $H'$;
7      foreach $h \in \mathcal{H}'$ do
8          if $avgDiv(h) > avgDiv(H^*)$ then
9              $H^* \leftarrow h$;
10         end
11     end
12     foreach $h \in \mathcal{H}'$ do
13         calculate necessary upper bounds of $h$ and prune $h$ based on $H^*$ ;
14     end
15     sort $h$ in the pruned $\mathcal{H}'$;
16     foreach $h \in \mathcal{H}'$ do
17         foreach $v \in V(h)$ do
18             advEnum ($h \setminus \{v\}$);
19         end
20     end
```

The advanced algorithm Algorithm 4 shows the algorithm with the discussed optimisations. Different from Algorithm 3, it has an extra loop (Lines 8 to 13) to efficiently prune connected *k*-core with least cost; after that, search order optimisations are applied (Line 18).

The correctness of Algorithm 4 is immediate given the correctness of Algorithm 3 and the correctness of upper bounds discussed.

The experiments are detailed in Section 6.2.

5 Attribute diversified community with maximum cohesiveness

In this section, we present the following motivating example to show how limiting the attribute diversity will effect the structure cohesiveness of the community. Then, we introduce an attribute diversified community search work that aims to maximise the structure cohesiveness while maintaining attribute diversity to a certain level. Lastly, we introduce the community model and search problem, before moving on to discuss the solutions.

Motivating example In Figure 6, a user of *Facebook* communicates in at least one or more languages like, English (E), French (F), German (G), Mandarin (M), etc. We consider a scenario, where we need to find a multi-lingual set of users to help marketing a product(s). The product(s) can be marketed across pages/groups of different languages to have a broader and diverse customer reach. The acquaintance between the users will help in marketing the product as they can convey their idea with ease and simultaneously share them through their mutual connections. The diversity among the users makes the community diverse in terms of the maximum number of languages shared, i.e., strong attribute diversification. Therefore, we need to find a community that is attribute diverse to a certain extent but maximises structural cohesiveness also. The *attribute diversified community with maximum cohesiveness model* can be used to find a community that maximises structural

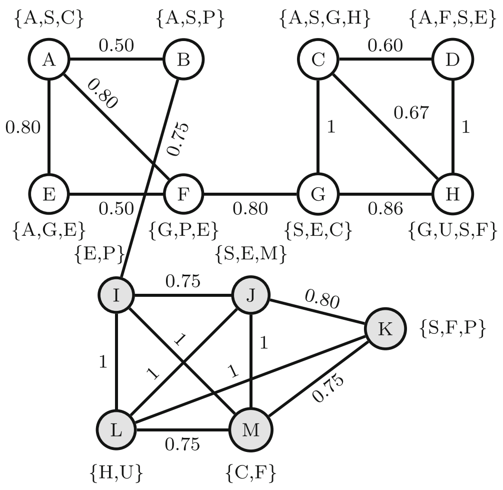


Figure 6 A Facebook graph

cohesiveness, while being diverse enough from the prospective of languages that a user can interact with. Each user is also connected to at least 2 other users to maintain a certain degree of structural cohesiveness.

The novel community model introduced considers attribute diversification as a constraint while maximising the structure cohesiveness as the primary search objective. In this model, the structure cohesiveness is based on the widely used k-core model, where each user inside the community has the maximum number of at least k number of connections.

In Figure 6, the attribute diversified community with maximum cohesiveness formed is by limiting the attribute diversity to 0.75 which is the subgraph induced by {I,J,K,L,M}, where each user knows at least 3 other users in the community, shown as shaded vertices in Figure 6. This implies that there are no other users in the community that can be more structurally cohesive for this level of attribute diversification.

This section is devoted to the study on how to model and search the attribute diversified community with maximum cohesiveness. We first define the diversity measure between every user of the community to capture the attribute diversification. Therefore, we define subgraph diversity measure, designed to find a community which maximises the structure cohesiveness. Based on the subgraph diversity measure, we model the attribute diversified community with maximum cohesiveness using attribute diversification as a constraint. We develop two efficient indices for real-time solutions, namely the CD-index, based on the hierarchical core decomposition and the novel advanced KD-index, a modified hierarchical core index stored at different intervals of τ. We then develop an efficient exact algorithm by proposing effective strategies for pruning the search space and implementing two heuristic search orders on vertices. We also propose pre-pruning techniques for reducing the input data.

5.1 Problem definition

In this section we introduce an attribute diversified community search work that focuses on finding a community maximising the structure cohesiveness while maintaining the attribute diversity to a certain level. The community model is discussed firstly. Then, the solutions are presented thereafter.

The notations specific to this section only are summarised in Table 2.

We formally define the subgraph diversity below.

Definition 9 Subgraph diversity, $\tau(H)$. Given a subgraph $H \subseteq G$, we define its *subgraph diversity* as:

$$\tau(H) = min\{div(u, v) \geq \tau | \forall u, v \in V(H), u \neq v\}$$

Table 2 Summary of notations used in this section

Notation	Definition
τ	the user given diversity threshold
$\tau(H)$	the subgraph diversity threshold of H
D	the corresponding diversity subgraph of H
H_k	the k-core set of G
R, P, X	the vertex sets
k^*	the optimal social core number found

Definition 10 (k, τ)**-core**. Given a subgraph $H \subseteq G$, a user given attribute diversity threshold τ, H is a (k, τ)-core, if H satisfies the following constraints simultaneously:

- *Connectivity*: H is connected;
- *Structure cohesiveness*: $\forall v \in V(H), deg(v) \geq k$;
- *Subgraph diversity*: $\tau(H) \geq \tau$.

Research Problem. ***Attribute diversified community search with maximum cohesiveness (ADC-MC)***. *Given an attributed graph G, a user given attribute diversity threshold τ, find a (k, τ)-core $H \subseteq G$ such that k is maximum.*

We introduce the following theorem based on the above definition of a (k, τ)-core.

Theorem 1 *ADC-MC of a graph G can be found inside any maximal connected k-cores of G if they exist.*

Proof From Definition 1, the vertices that are not part of any maximal connected k-cores clearly cannot meet the structural cohesiveness requirement. Thus, the ADC-MC of a graph G can only be found inside any connected maximal k-cores of G, which maximises the structure cohesiveness while satisfying the diversity constraint τ. □

Before moving forward, we introduce the following definition and present an observation as follows.

Definition 11 Diversity Graph. Given a diversity threshold τ, let D denote a new graph named *diversity graph* with $V(D) = V(G)$ and $E(D) = \{(u, v) \mid div(u, v) \geq \tau$ and $u, v \in V(G)\}$.

Observation 3 *Given a (k, τ)-core H, the $V(H)$ induced subgraph of D is a complete subgraph (i.e., a clique).*

Discussion On analysing the process to find the most cohesive (k, τ)-community, we first find all the maximal cliques in D, and then for each clique we use k-core decomposition to arrive at the largest k. The bottleneck lies in initially enumerating a large number of redundant maximal cliques to get the optimal result. Hereafter, to overcome the above bottleneck, we propose two novel indices to answer our query in optimal time. Further, we introduce local search order heuristics to speed up the clique enumeration to find the optimal k denoted by k^* and the corresponding (k^*, τ)-core denoted by H^*.

Now, we first introduce a baseline index. This baseline index only saves vertices that are structurally promising. Then, we introduce two heuristics to speed up the search for finding k^*. Later on, we propose a novel advanced index structure that can help us to identify the vertices that are both structure cohesive and attribute diversified. The advanced index improves the efficiency to find the optimal result as demonstrated later in the experiments.

5.2 Baseline approach

The idea here is to organise the vertices in a hierarchical manner which helps to speed up the baseline algorithm. We use a core based heuristic, where vertices of the larger core are prioritised over the vertices in the smaller core. The heuristic terminates the search early if the core number of the vertex explored is smaller than the largest k found so far.

Index construction Given a graph G, the baseline index organises the vertices $V(G)$ based on the k-core subgraph with different k. The baseline index contains all the vertices in $V(G)$ and each vertex is contained in only one k-core tree node.

By using the nested property of core decomposition [4], we build the pre-computed hierarchical index, denoted by CD-Index in $\mathcal{O}(m)$ time. The index is saved in a space and cost effective hierarchical manner.

Given a maximal connected k-core set of graph G, we construct the k-core tree. The k-core tree contains all the vertices in $V(G)$, and each vertex is exclusively contained in one k-core tree node, where the coreness of the vertices is k. The structure follows a parent-child relation between the two nodes, i.e., parent k-core tree node and child k-core tree node. The creation of the relation between the expected parent-child k-core tree nodes follows the containment relation of the k-core components. Thus, the k-core tree formed for each connected k-core set represents the hierarchical nature of core decomposition.

If we have a disconnected graph G, the index construction can be done for each k-core set in G.

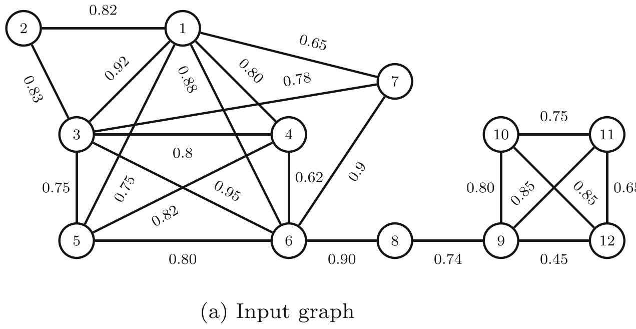
The core based heuristic used to retrieve a maximal k-core set in linear time reduces the time complexity significantly.

Example In Figure 7a, we apply arbitrary values to the edges and assume both the social graph H and its diversity graph D are the same. We precompute CD-Index as shown in Figure 7b. We see vertices of each tree node corresponding to a particular k. Here, the k-core tree structure constructed maintains the parent-child relationship between the k-core nodes.

Algorithm 5 baseKDAlgo.

```
Input: CD-Index of $G$, diversity threshold $\tau$
Output: $H^*$ : the most cohesive $(k^*, \tau)$-core community for $k^*$
1  $H^* \leftarrow \phi, k^* \leftarrow 0$;
2  $k_{max} \leftarrow$ maximum social core number from CD-Index;
3  BasicADC-MC($k_{max}, k^*, \tau$);
4  return $H^*$;
5  Procedure BasicADC-MC ($k_{max}, k^*, \tau$)
6      foreach $k \in range(k_{max}, 2) \mid k > k^*$ do
7          $\mathcal{H}_k \leftarrow k$-core set from CD-Index;
8          foreach $h \in \mathcal{H}_k$ do
9              delete the social edges in $h$ with diversity less than $\tau$ and find the
               maximal social core number $k'$;
10             if $k' \geq k^*$ then
11                 $H \leftarrow$ connected $(k', \tau)$-core subgraph;
12                 $D \leftarrow$ diversity subgraph of $H$;
13                 $H^* \leftarrow$ bkPivotMod($\phi, V(D), \phi, H^*$);
14             end
15         end
16     end
```

Algorithm 5 shows the process of finding k^* for which a (k^*, τ)-core can exist for a given diversity threshold τ. Initially at line 2, the maximum social core number k_{max} from the CD-Index is identified and the BasicADC-MC subroutine is called in Line 3. The BasicADC-MC



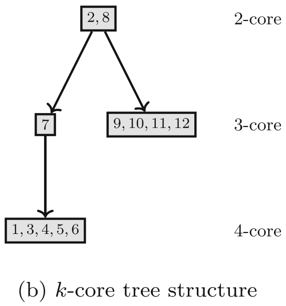


Figure 7 Example for attribute diversified community with maximum cohesiveness

subroutine enumerates all the possible k-core set to find k^* in Lines 5 to 16. The subroutine does this by retrieving a k-core set $\mathcal{H}_k$ from CD-Index in Line 7, and then exploring each connected k-core set $h \in \mathcal{H}_k$ in Lines 8 to 15. It continues exploring as long as the k values of the core sets are greater than k^* found so far. In doing so, the search space is limited only to the current core set.

We present two heuristic approaches based on the vertex search orders irrespective of the social core number. These heuristics are based on the diversity core number for the vertices of the diversity graph D of H. We explore and show their effect on the efficiency of the search space enumeration for finding the optimal result.

Algorithm 6 bkPivotMod(R, P, X, H^*).

```
1  if ($P \cup X = \phi$) then
2      $k' \leftarrow kCoreCheck(R)$;// compute the maximum social core
           number of $R$
3      $H' \leftarrow$ optimal ($k'$, $\tau$)-core community in $R$;
4      if $k' < k^*$ then continue;
5      $k^* \leftarrow k'$;
6      $H^* \leftarrow H'$;
7  end
8  choose a pivot vertex u in $P \cup X$ to maximise $|P \setminus N(u)|$;
9  foreach ($v \in P \setminus N(u)$) do
10     bkPivotMod ($R \cup \{v\}$, $P \cap N(v)$, $X \cap N(v)$, $H^*$);
11     $P \leftarrow P \setminus \{v\}$;
12     $X \leftarrow X \cup \{v\}$;
13 end
```

Pivot selection After retrieving $\mathcal{H}_k$ (Line 6, Algorithm 5) from CD-Index. For each connected subgraph k-core set $h \in \mathcal{H}_k$, we use an implementation of bk algorithm [6], using a pruning technique technique called *pivoting* [26]. We modify this bk-based pivoting algorithm according to our problem needs. The pivot based bk algorithm, bkPivotMod implementation is given in Algorithm 6. We have a social core number check whenever a clique is generated (Line 2, Algorithm 6) to further optimise k^*. If optimised, k^* will then act as a lower bound to prune out insignificant search branches later (Line 9, Algorithm 5). This check ensures, a maximal social core number will exist that satisfies both, maximising the structure cohesiveness *(our objective)* by satisfying the diversity threshold. The modified bk-based pivoting implementation is given in Algorithm 6.

We can further improve the search order mechanism of bkPivotMod algorithm by introducing a heuristic based on the degeneracy order of the vertices in D of H.

Using degeneracy order with pivot selection

Algorithm 7 bkPivotDeg($V(D), k^*$).

```
1  $\delta(D) \leftarrow$ the degeneracy order of $V(D)$;
2  **foreach** ($v_i$ *in a degeneracy ordering of* $v_1, v_2, v_3, \cdots \in V(D)$) **do**
3      **if** ($\delta_{v_i} \leq k^*$) **then**
4          $P \leftarrow P \setminus \{v_i\}$;
5          $X \leftarrow X \cup \{v_i\}$;
6          $continue$;
7      **end**
8      bkPivotMod ($R \cup \{v_i\}, P \cap N(v_i), X \cap N(v_i), H^*$);
9      $P \leftarrow P \setminus \{v_i\}$;
10     $X \leftarrow X \cup \{v_i\}$;
11 **end**
12 **return** $H^*$;
```

We use the degeneracy order when evaluating vertices in the diversity graph D of H, which can nicely bound the search depth of the promising subgraph to the degeneracy of the subgraph. Generally, the degeneracy of the subgraph is much smaller than the total number of vertices contained in the subgraph.

The degeneracy based bk-pivoting algorithm, bkPivotDeg implementation is given in Algorithm 7. The implementation of degeneracy based heuristics uses the bkPivotMod algorithm as a subroutine to further improve the efficiency of the search order enumeration [16, 17]. We limit the search branch to only enumerate the vertices in the degeneracy ordering as long as the vertex degeneracy is greater than optimal k found. In this way we restrict and reduce the search space significantly.

Also, we can modify the bkPivotMod algorithm by further adding a structure constraint check after choosing a pivot vertex u, Line 8 of Algorithm 6. This check helps stop generating candidate maximal cliques not satisfying the constraint. These candidate maximal cliques are represented by the vertices of $R \cup P$, where R is a partial clique which may contain the optimal result, and P is the set of candidates vertices to further expand R. We present the following theorem based on the above discussion.

Theorem 2 *The enumeration stops if $R \cup P$ induced social subgraph does not satisfy the structure constraint.*

Proof For every vertex $u \in P \cup X$, a vertex u can only be added to the current optimum result R if it is in the $R \cup P$ induced social subgraph such that $k > k^*$. This is because the vertex u that is added from P to R should at least be a connected induced k'-core social subgraph. Therefore, the vertex $u \in R \cup P$ may be discarded in the following search due to the structure constraint check. This structure constraint check works as an early termination condition to further reduce the search space significantly. □

Apart form the above heuristics, we further optimise the search order enumeration by reusing the computation incurred for implementing Theorem 2.

1. If $V(R \cup P)$ is the same as its child $V(R_c \cup P_c)$ during a recursion, then we can skip the social core number constraint check as the maximal social core number remains the same.
2. If the social core number of $R_c \cup P_c$ is less than its parent social core number $R \cup P$, then we can terminate the exploration branch as we can not further optimise k'.
3. We can further skip the social core number constraint check, when $V(P_c \cup X_c)$ is empty, and $V(R_c \cup P_c)$ is the same as its parent $V(R \cup P)$ because both P and P_c are empty.

Example In Figure 7a, suppose a query comes for τ=0.65. Using the pre-computed CD-Index, we directly get the corresponding maximal k-core subgraph for k_{max}=4, i.e., the subgraph induced by {1, 3, 4, 5, 6}, which is also our optimal result for k^*=4.

Correctness The correctness of Algorithm 5 is based on the correct construction of the heuristic core based CD-Index, the correctness of state of the art enumeration algorithm [6, 16, 17] and correct computation of the structure constraints [39]. Algorithm 5 correctly finds the maximal k by returning the most cohesive (k, τ)-core.

Space complexity During the index construction phase, the vertices are saved only once corresponding to their respective k-core tree node. Therefore, the space cost of the index is bounded by $\mathcal{O}(n)$. Also, to save a set of edges cost $\mathcal{O}(m)$ and the adjacency list of each node cost $\mathcal{O}(m + n)$. Hence, the total space cost is bounded by $\mathcal{O}(m + n)$.

Time complexity For finding the optimal result, retrieving a set of k-core vertices from the index takes $O(V(H_k))$ time. The time complexity of the state of the art enumeration algorithm [6] is equal to the number of cliques generated but due to the structure constraints check and the local search order heuristics presented later, the time complexity is reduced significantly.

5.3 Advanced KD-index

The core based heuristic CD-Index only considers prioritising vertices that are structurally promising. Since the definition of the (k, τ)-core considers both the structure and the attribute properties, a heuristic that prioritises vertices that are promising from both structure cohesive and attribute diversified perspectives would be better.

Accordingly, we propose a novel index structure, KD-Index to help us identify the vertices that are promising both from structure cohesive and attribute diversified perspectives. We make the index more general for different queries by pre-computing promising vertices for different diversity thresholds. We term each distinct diversity thresholds as a *diversity threshold point* denoted by Ψ. We denote them as $\mathbb{T} = \{\Psi_1, \ldots, \Psi_N\}$, where $\mathbb{T}$, a set of diversity threshold points spaced at equal intervals within (0, 1) that follows a total order, and N is the total number of diversity threshold points. For any 2 diversity threshold points Ψ_i, Ψ_j such that for any $i < j$ and $i \neq j$, $\Psi_i < \Psi_j$ follows a total order.

Index construction Given a graph G, the advanced index organises the vertices $V(G)$ for each diversity threshold point $\Psi \in \mathbb{T}$. Each Ψ will have its own hierarchical k-core tree structure, such that the diversity value of the edges of the promising vertices will satisfy the corresponding Ψ value. We do this by deleting all the edges whose diversity values are less than current Ψ; i.e., $div((u, v)) < \tau$, and then do k-core decomposition to create the corresponding k-core tree. We repeat this process for every diversity point $\Psi \in \mathbb{T}$. Thus, a k-core tree will be created for every $\Psi \in \mathbb{T}$. Here, every k-core tree will have the same structure as that of the baseline index CD-Index. The same way, every k-core tree will be divided into k-core tree nodes. Every promising vertices will belong to a specific k-core tree node. Therefore, each Ψ will have its own unique hierarchical k-core tree.

So whenever a query comes, we first identify the diversity threshold point Ψ from the set $\mathbb{T}$ such that Ψ is just smaller than τ. In this way, we can retrieve the reduced k-core subgraph from the k-core tree corresponding to the current Ψ. In this way, we significantly reduce our search space from redundant enumeration. We can deduce the following observation:

Observation 4 *The optimum k^* can only be found in a connected k-core subgraph corresponding to the current Ψ, if it exists when the enumeration starts.*

Using the above observation, the major advantage of the advanced index is that we reduce the size of the retrieved subgraph. This reduces the search space enumeration to get the optimal result.

Example In Figure 8a-c, we pre-compute KD-Index at 3 diversity threshold points, i.e., $\mathbb{T} = \{0.25, 0.5, 0.75\}$ and vertices of each tree node corresponding to a particular k for respective Ψ's. Again, for a query τ=0.65, now using the pre-computed advanced KD-Index, we first identify the Ψ just below 0.65. In this case it is 0.50. We directly get the corresponding maximal k-core subgraph for maximal k=4, i.e., the subgraph induced by $\{1, 3, 4, 5, 6\}$, as shown in Figure 8b.

Another advantage of the index is when an optimal k^* is not encountered for the current Ψ, we terminate the algorithm. Because, if we can not get a feasible solution for the current Ψ, then we definitely cannot get a feasible solution for any diversity threshold point greater than the current Ψ.

Space complexity For the advanced index, vertex set is saved at most $\mathcal{O}(|\mathcal{T}| \cdot |V(G)|)$ in the worst case, where $|\mathcal{T}|$ is the total number of diversity points and each vertex is saved only once for a specific diversity point. Hence, the total space cost is bounded by $\mathcal{O}(|\mathcal{T}| \cdot |V(G)|)$.

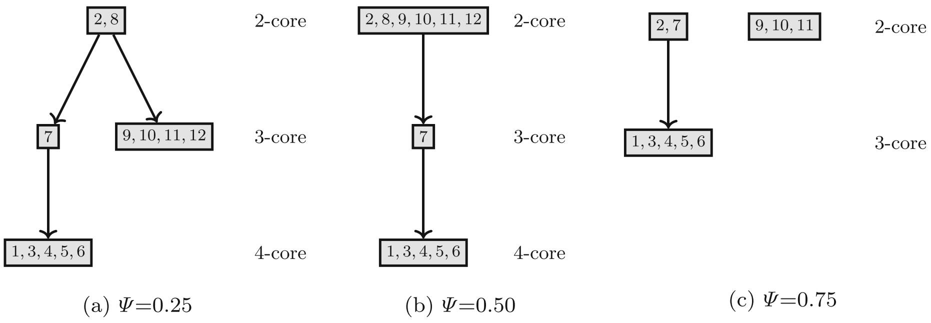


Figure 8 Tree structure for different diversity threshold points

5.4 Advanced solution for finding most cohesive (k, τ)-core

With the advanced KD-Index, we can optimally retrieve the vertex set by identifying the Ψ_p threshold point by only visiting the vertices of the reduced (k, Ψ_p)-core set $\mathcal{H}_k$. Algorithm 8 shows the pseudo code for finding the maximal k.

Then, we find the first diversity threshold point Ψ that is just smaller than the given parameter τ in Line 2. Corresponding to the current Ψ, we identify the maximum social core number k_{max} from the KD-Index in Line 3. Then, we call a subroutine EffiADC-MC to start our enumeration in Line 4.

The EffiADC-MC subroutine works the same way as the BasicADC-MC subroutine of Algorithm 5 but with an improved vertex degeneracy based enumeration algorithm bkPivot-Deg in Line 14. The bkPivotDeg algorithm is presented in Algorithm 7. As previously said, the vertex degeneracy order bounds the depth of the promising subgraph to the degeneracy of the subgraph. Also, it uses the pivoting mechanism bkPivotMod to further improve the enumeration.

Algorithm 8 advKDAlgo.

```
   Input: KD-Index of $G$, diversity threshold $\tau$, $\mathbb{T}$ set of diversity threshold points
   Output: $H^*$ : the most cohesive $(k^*, \tau)$-core community for $k^*$
1  $H^* \leftarrow \phi, k^* \leftarrow \phi$;
2  Identify $\Psi$ from $\mathbb{T} = (\Psi_1, \dots, \Psi_n)$ such that $\Psi$ is just smaller than $\tau$;
3  $k_{max} \leftarrow$ maximum social core number from KD-Index corresponding to $\Psi$;
4  $H^* \leftarrow$ EffiADC-MC($k_{max}, k^*, \tau$);
5  return $H^*$;
6  Procedure EffiADC-MC ($k_{max}, k^*, \tau$)
7      foreach $k \in range(k_{max}, 2) \mid k > k^*$ do
8          $\mathcal{H}_k \leftarrow$ set of $(k, \Psi)$-core set from KD-Index;
9          foreach $h \in \mathcal{H}_k$ do
10             delete the edges in $h$ with diversity less than $\tau$ and find the maximal
               social core number $k'$;
11             if $k' \geq k^*$ then
12                 $H \leftarrow (k', \tau)$-core subgraph;
13                 $D \leftarrow$ diversity subgraph of $H$;
14                 $H^* \leftarrow$ bkPivotDeg($V(D), k^*$);// Implements
                       pivot-based bk algorithm using degeneracy
                       order
15             end
16         end
17     end
```

Correctness The correctness of Algorithm 8 is based only on the correct construction of the advanced index KD-Index and correctly computing Theorem 2. Also, the state of the art enumeration algorithm and computation of the structure constraints are the same as before.

Complexity During the advanced index construction phase the vertices are saved only once corresponding to their respective core numbers at each interval. Therefore, the space cost of the index is bounded by $\mathcal{O}(n \cdot |\mathcal{T}|)$, where n is the number of vertices in G and $|\mathcal{T}|$ are the number of diversity threshold points. For finding the optimal result, retrieving a set of vertices from KD-Index costs $\mathcal{O}(V(H_k))$ time corresponding to the diversity threshold point Ψ such that Ψ is just greater than τ, and the time cost for further enumeration is again equal to the number of cliques generated during the recursion. The clique enumeration is reduced significantly using the local order degeneracy heuristic and structure constraints checks.

The experiments are detailed in Section 6.3.

6 Experiments

All the experiments for the 3 problems were conducted on a PC with 2.4GHz Intel Core i7 (6-core) processor, 16GB RAM, running Windows 10. All the algorithms were implemented in C++ with -O3 optimisation level using Microsoft VSCode with MinGW 64-bit compiler.

6.1 Attribute diversified community search - max-min problem

Implemented algorithms We implemented the algorithms proposed in Sections 3.2 and 3.3. Algorithms 1 and 2 are denoted as Alg-1 and Alg-2 correspondingly in this section. For extra performance evaluations, we also extended our algorithms to finding *top-l results* and reported the performance.

Datasets The six real datasets include Facebook_0, Twitter, Brightkite, Deezer, Gowalla and DBLP, that are obtained from *snap.standford.edu* and *konect.uni-koblenz.de*. The data were cleaned by removing all self loops. For Facebook_0 and Twitter, the diversity values between the relationships were pre-computed using Jaccard similarity based on the attributes of vertices. For Brightkite, Deezer, Gowalla and DBLP, random values lying between (0,1) were generated between the relationships of the edges. Table 3 presents the statistics for all datasets.

Parameter settings The experiments were evaluated using different settings of query parameters: k (the minimum core number) and l (top-l results). The ranges of the parameters and their default values are shown in Table 4, in which we select reasonable k and l based on datasets.

6.1.1 Efficiency evaluation

Varying *k* Figure 9 shows results for different values of k when l has the default values. The figure clearly states the results that Alg-2 performs better than Alg-1 because of its

Table 3 Real world datasets used for max-min problem

Datasets	Vertices	Edges	d_{max}	d_{avg}
Facebook	333	2,519	77	7.56
Twitter	344	3,362	100	9.77
Brightkite	58,228	214,031	1,134	7.3
Deezer	143,884	846,897	420	11.77
Gowalla	196,595	947,059	14,730	9.67
DBLP	317,080	1,049,866	343	6.62

Table 4 Parameter settings for the datasets used in max-min problem

Datasets	Default k	Default l	Range of k	Range of l
Facebook_0	6	14	3,4,5,6,7,8,9	10,12,14,16,18,20
Twitter	18	50	12,14,16,18,20,22	20,40,60,80,100,120
Brightkite	16	20	10,12,14,16,18,20,22	10,20,30,40,50
Deezer	15	60	8,10,12,14,16,18,20	40,60,80,100,120
Gowalla	22	30	15,20,25,30,35,40	10,20,30,40,50
DBLP	32	125	10,15,20,25,30,35	100,150,200,250,300

almost linear time complexity. While Alg-2 is performing, the size of the graph progressively decreases on edge deletions and thus the search space reduces. On the other hand, as we increase the value of k, the run-time of both becomes nearly the same. For Alg-1, it performs poorly due to its quadratic time complexity. For a large dataset like DBLP, we see that for values of $k = 10, 20$, Alg-2 significantly outperforms Alg-1 and as we increase k, both of them become almost linear while Alg-2 still performs better. On the other hand for small and medium datasets like Facebook, Twitter, Gowalla, Brightkite and Deezer, we see that Alg-2 performs better for all values of k.

Varying *l* As shown in Figure 10, we vary the parameter l for fixed k as the default values for different datasets. For Facebook and Twitter datasets, the infinite times are set at 1 and 100 secs respectively. Over all the datasets, the processing time to get the *top-l* diversified communities for both algorithms, Alg-1 and Alg-2 tend to remain the same while Alg-2 still beats Alg-1. In the Twitter dataset, for the values of l=20, 40, 60, Alg-1 finds the top-l results beyond the infinite time threshold that we set to be 100 secs. In contrast, for the values of l= 80, 100, 120, Alg-1 finds the results in 97.56, 94.35 and 83.28 seconds respectively. From all the figures, we can clearly make out that Alg-2 is 100-300% faster than Alg-1 on all the datasets. Both of the results tend to remain the same on varying l because the l^{th} value increases on subsequent edge deletions in top-l results.

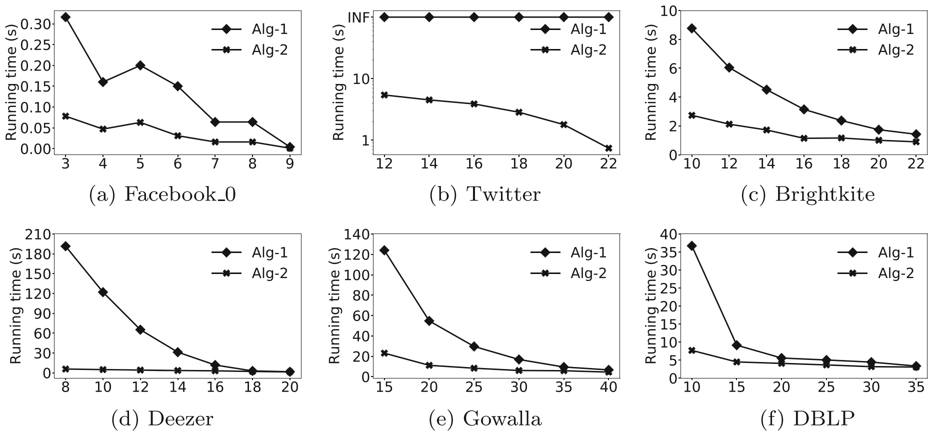


Figure 9 Varying k for the datasets used in max-min problem

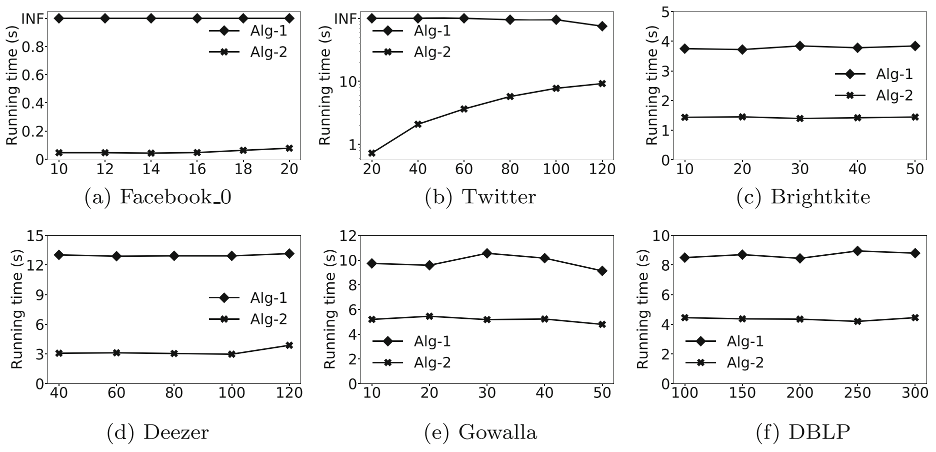


Figure 10 Varying l for the datasets used in max-min problem

Scalability We vary the size of the graph to see the scalability of Alg-1 and Alg-2. We show the results in Figure 11. It shows that Alg-2 does not get affected for finding the most diversified community and thus remains almost constant. But, Alg-1 evidently shows that its time increases with the increasing size of the graph.

6.1.2 Effectiveness evaluation

A case study on IMDb was conducted to demonstrate the effectiveness of our proposed max-min model. This dataset contains 5090 nodes and 8577 edges. We used the dataset at 12% scalability by selecting random edges. We specifically use this dataset to show the motivating example presented in Figure 3 for the max-average problem. We then pre-process the IMDb dataset to show an edge between two nodes, i.e., personalities, who have worked in no less than 5 films. Each vertex has 3 distinct keyword genres. Each genre is given an abbreviation such as *M* for *Music*, *D* for *Drama*, *B* for *Biography*, *Cr* for *Crime*, *Wr* for *War*,

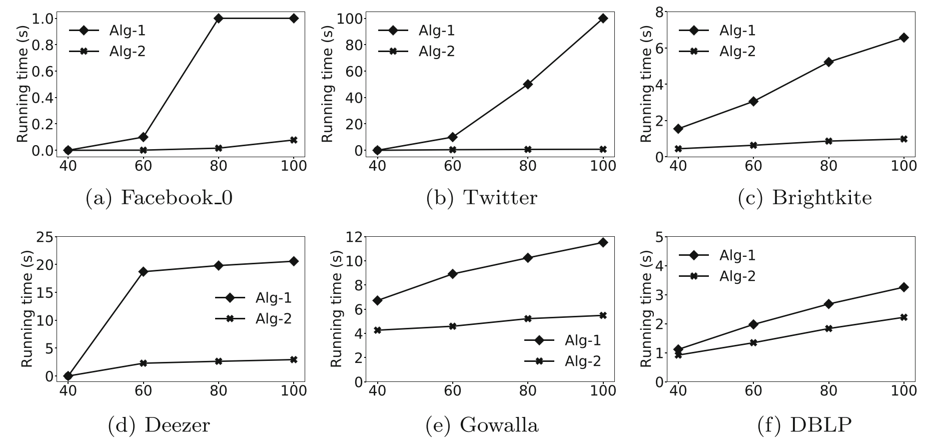


Figure 11 Scalability for the datasets used in max-min problem

C for *Comedy*, *R* for *Romance*, *SF* for *Science Fiction*, *Ac* for *Action*, *H* for *History*, *T* for *Thriller*, *S* for *Sport*, *Fm* for *Family*, *FN* for *Film Noir*, *N* for *News*, etc.

The community found by max-min model contains 11 personalities from Hollywood with their 14 social relationships that maximise the minimum attribute diversity. In the community, each personality is socially connected to at least 2 other personalities. The community expresses high attribute diversity as compared to a k-core model. This community has high diverse social connections between the personalities whose members form a diverse representative group.

In Figure 12, the result for the max-min model is illustrated for IMDb.

We conducted a case study on Facebook_0 dataset to demonstrate the effectiveness of our proposed max-min model. Compared to k-core, the max-min model allows us to find a group of diverse representative members whose members exhibit high diverse relationships. Figure 13 shows a case of Facebook_0 with $k = 6$. This community has 22 users with 100 diversified relationships.

In Figure 13, the result for the max-min model is demonstrated. First of all, the community found is still dense. Secondly, the found members have diverse relationships. The number of members in the resultant community is not large, however, they are representative.

6.2 Attribute diversified community search - max-average problem

Implemented algorithms We implemented all algorithms proposed in this paper. Algorithms 3 and 4 are denoted as Alg-3 and Alg-4 correspondingly in this section. All the algorithms have found the attribute diversified community maximising the max-average problem and have correspondingly reported the performance.

Datasets The four real datasets include Facebook, Brightkite, DBLP and ACM that are obtained from *snap.standford.edu* and *konect.uni-koblenz.de*. The data were cleaned by removing all self-loops. The edge diversity values were pre-computed using Jaccard similarity. Table 5 presents the statistics for all datasets.

Parameter settings The experiments were evaluated using different settings of query parameters: k (the minimum core number) to detect results, i.e., most attribute diversified community. The range of the parameter k is shown in the last column of Table 5. All the experiments were conducted at least 5 times and their average was taken.

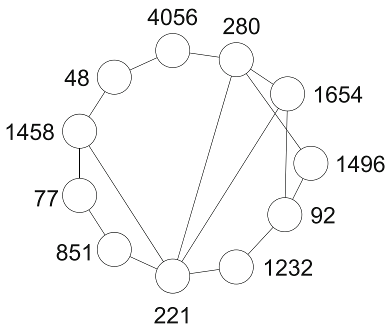


Figure 12 Case study on IMDb for k=2

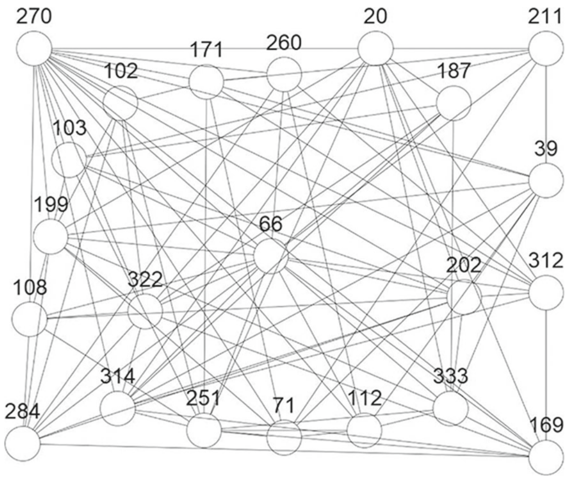


Figure 13 Case study on Facebook_0 for $k = 6$

6.2.1 Efficiency evaluation

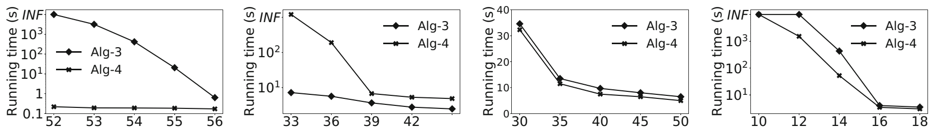
Varying k We vary the value of k for detecting results. The range of k for every dataset is chosen to select a diversified community with high cohesiveness. We report the results in Figure 14. For all the datasets, we see that Alg-4 outperforms Alg-3 for different values of k. This is because of the pruning effectiveness over the enumeration search space which is much higher for smaller values of k compared with large values of k. Because the size of maximal k-core is greater for small values of k, it makes bound check in Alg-4 more effective than Alg-3. But for larger values of k, the bound in Alg-4 becomes tighter by following Definition 4.1. For DBLP and ACM datasets, Alg-4 performs slightly better than Alg-3 by a magnitude margin of around 20%. Although Alg-4 performs better, the graph itself is very dense, which makes the bound check moderately effective for those value of k.

Scalability To see the scalability of the proposed Alg-3 and Alg-4 on varying the size of graph. We show the results for attribute diversified communities for different datasets with default k set at 55, 39, 40, 18 in order as presented in Table 5. Figure 15 shows that, Alg-4 performs better than Alg-3 in all the scenarios when varying graph sizes, with the main reason being the effective upper bound check and search order optimisation for Alg-4.

Also, since the density of the graph varies with different ratios, the pruning effectiveness of Alg-4 clearly outperforms Alg-3.

Table 5 Real world datasets used for max-average problem

Datasets	Vertices	Edges	d_{max}	Range of k
Facebook	3,892	17,237	62	52,53,54,55,56
Brightkite	58,228	214,031	1,134	36,39,42,45,48
DBLP	317,080	1,049,864	343	25,30,35,40,45
ACM	359,748	1,063,553	5,720	10,12,14,16,18



(a) Facebook (b) Brightkite (c) DBLP (d) ACM

Figure 14 Varying k for the datasets used in max-average problem

Pruning effectiveness evaluation We also evaluate the pruning effectiveness when varying k. This is evaluated by two measurements: the number of recursions used by Alg-4 over the number of recursions used by Alg-3, denoted by Alg-4/Alg-3, and the percentage of the total search space explored by Alg-4 for finding the attribute diversified community results, denoted by Alg-4/theo. Here, theo means the total number of theoretical recursions. The results are shown in Figure 16. The search space explored by Alg-4 for both Facebook and Brightkite is around 50%, for DBLP is around 40% and for ACM is around 25% as compared to Alg-3. This clearly justifies the pruning effectiveness of the upper bound and search order optimisation. On the other hand, Alg-4 only needs to explore very small percentage of the possible search space to get the most diversified result, i.e., between 10-20% on average.

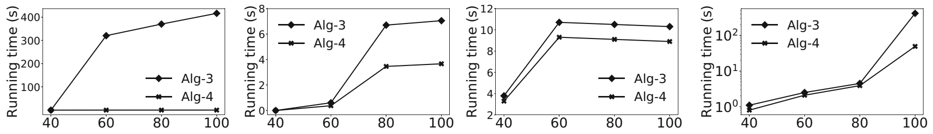
6.2.2 Effectiveness evaluation

We use the IMDb dataset with the same settings as described in Section 6.1.2 to demonstrate the effectiveness of our max-average model.

The community found by the max-average problem contains 28 film personalities and 44 diversified relationships from the Hollywood film industry that are connected with minimum degree of at least 2. Compared to k-core, the max-average model allows us to find meaningful communities whose relationships exhibit diversity in terms of interested attributes. This community expresses the wide depth across different genres and helps to find connected personalities who have worked together more often. In Figure 17, the result for the max-average problem is illustrated for IMDb.

We conducted a case study on Facebook_0 dataset to demonstrate the effectiveness of our proposed max-average model. Compared to k-core, max-average model allows us to find meaningful communities whose members exhibit difference in terms of interested attributes. Figure 18 shows a case of Facebook with $k = 6$. This diversified community has 35 users with 235 relationships as compared to the community found in Figure 13 where a representative community of users is found.

As you can see for both the case studies, the community found by the max-average problem encompasses more nodes than the community found by the max-min problem because it includes a number of nodes with less attribute diversity but they contribute more to the



(a) Facebook (b) Brightkite (c) DBLP (d) ACM

Figure 15 Scalability for the datasets used in max-average problem

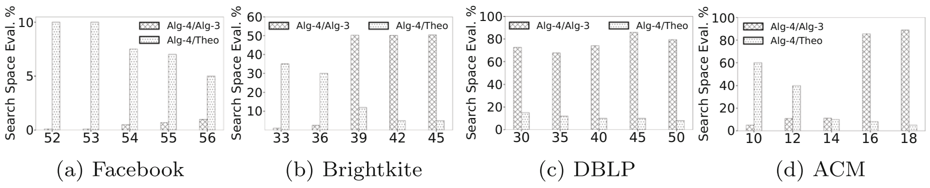


Figure 16 Pruning effectiveness for varying k for the datasets used in max-average problem

Figure 17 Case study on IMDb for k=2

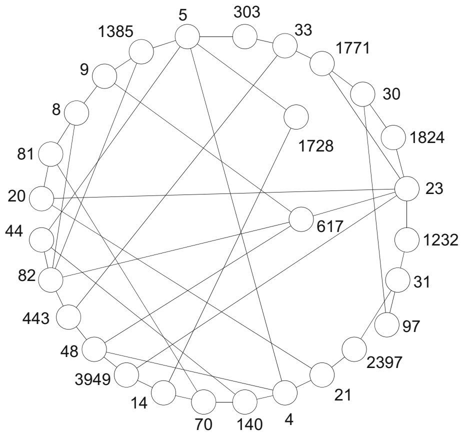
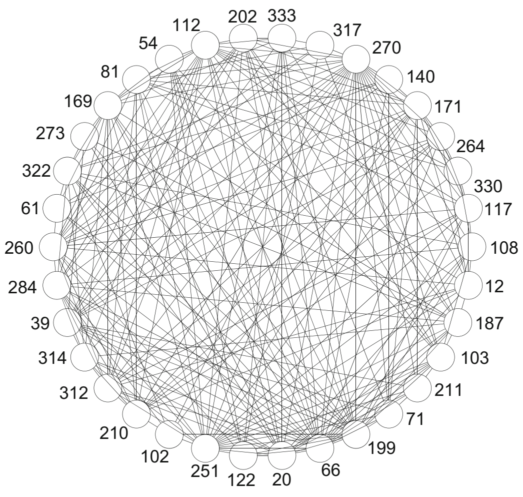


Figure 18 Case study on Facebook_0 for k=6



overall diversity of the community and leaving them out will cause them to lose the structure constraints.

6.3 Attribute diversified community search with maximum cohesiveness

Implemented algorithms Here is the description of the different algorithms used in the experiments part. Alg-A denotes the combination of baseline hierarchical k-core based index, Algorithm 5 with pivot based bk-algorithm, Algorithm 6. Alg-B denotes the combination of baseline hierarchical k-core based index, Algorithm 5 with pivot based bk-algorithm using the optimisations and local degeneracy order of the vertices, Algorithm 7. Alg-C denotes the combination of advanced index, Algorithm 8 with pivot based bk-algorithm, Algorithm 6. Alg-D denotes the combination of advanced index, Algorithm 8 with pivot based bk-algorithm using the optimisations and local degeneracy order of the vertices, Algorithm 7.

Datasets The six real datasets include Facebook, Twitter, out_arenas, Brightkite, DBLP and ACM datasets are obtained from *snap.standford.edu* and *konect.uni-koblenz.de*. Facebook and Twitter datasets have their users and real keyword attributes anonymised to calculate the diversity values. Facebook dataset only uses language attributes and users are deleted if they do not have language attributes. Whereas for other datasets between every pairwise nodes, synthetic attributes are randomly generated for real values between (0, 1) which are used as diversity values. These values are once generated and saved along with the index for the entire run of the experiments to maintain consistent result. Different properties of each dataset are shown in Table 6.

Parameter settings Since the user usually does not need diversity values less than 0.50. We set our baseline CD-Index at this diversity threshold point. We then create the advanced KD-Index for 5 diversity threshold points, i.e., $\mathbb{T} = \{0.50, \ldots, 0.90\}$ to perform experiments. The user given diversity threshold τ is the input to all the algorithms along with the graph. k represents the optimal social core number achieved for diversity threshold τ.

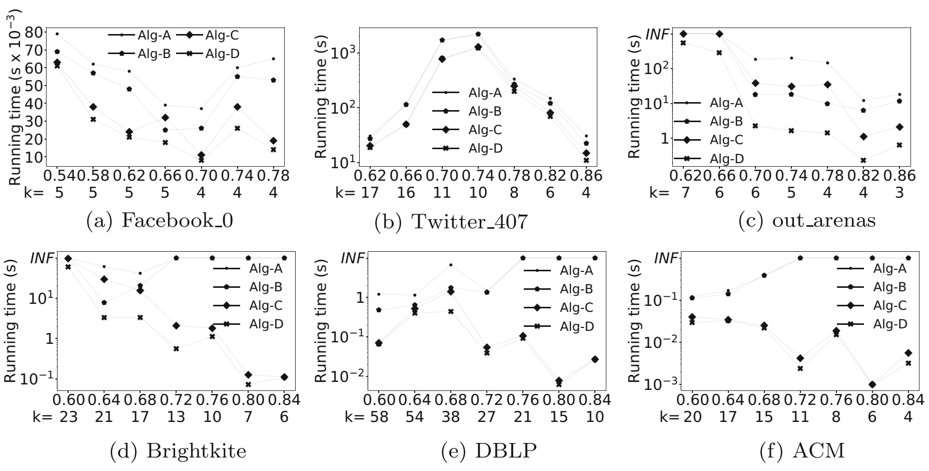
6.3.1 Efficiency evaluation

Overall running time of algorithms We evaluate the different algorithms by varying the user input query diversity threshold τ that lies between the range of {0.50-0.90} across all datasets.

Figure 19 shows the results of the algorithms based on the advanced index KD-Index which performs better than the algorithm based on the basic k-core index CD-Index. The

Table 6 Real world datasets

Datasets	Vertices	Edges	d_{max}	d_{avg}	Size (MB)
Facebook_0(Fb_0)	333	2,519	77	7.56	0.054
Twitter_407	344	3,362	100	9.77	0.044
out_arenas	10,647	24,315	205	2.27	0.67
Brightkite	58,228	214,031	1,134	7.30	2.90
DBLP	317,080	1,049,866	343	6.62	14.94
ACM	359,748	1,063,553	5,720	2.95	15.99



(a) Facebook_0 (b) Twitter_407 (c) out_arenas
(d) Brightkite (e) DBLP (f) ACM

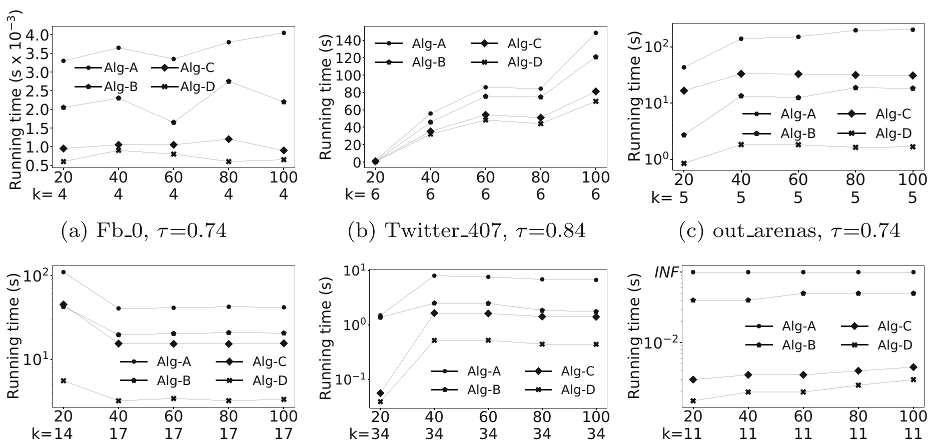
Figure 19 Overall running time for different τ

overall running times for Alg-C and Alg-D decrease since they are based on the KD-Index, which reduce the search space enumeration to get the result. Also the advantage of the KD-Index is quite evident over the CD-Index. For some datasets, Alg-A and Alg-B cannot even enumerate and hence achieve the respective infinite running time values set for the each dataset for some values of τ. In Twitter_407 dataset, for some low values of τ, Alg-C performs almost the same as Alg-D, but as τ increases, Alg-D performs better than Alg-C. Twitter_407 is denser than others and the advanced optimisation based on the degeneracy order of the vertices does not play a significant role. In comparison for all the other datasets, Alg-D outperforms Alg-C by the orders of magnitude of at least 100 times in out_arenas, 10-100 times in Brightkite, 2-5 times in DBLP and ACM.

Next, when we increase the user input query diversity threshold τ, the maximum k for which we find the most cohesive diversified community also decreases in an almost linear pattern. For some intermediary values, all the algorithms take slightly more time than the previous values, which is because on decreasing k the size of the k-core subgraph under consideration is the largest than the previous larger k found (as a k-core lies in a $(k+1)$-core, because of the hierarchical nature of core decomposition).

Scalability We evaluate the scalability of the algorithms by randomly selecting 20%, 40%, 60%, 80%, 100% vertices in each dataset. For a given user input query diversity threshold τ, we compare the time taken to get the optimal k for different datasets. Figure 20 shows the results for different datasets. Generally, Alg-C and Alg-D based on the KD-Index perform better than Alg-A and Alg-B based on CD-Index. For Brightkite dataset with 20% vertices, it takes more time as the most cohesive diversified community exists for k=14 in comparison to k=17, which we get for vertex sizes greater than 20% of the dataset. Therefore, extra cost is incurred as the size of the k-core subgraph existing in 20% of the dataset which is more than the rest enumerated afterwards.

Time cost of index construction We evaluate the time taken to create the indices for different datasets. We also vary the number of diversity values (default is 5, represented by $|\mathbb{T}|$



(a) Fb_0, τ=0.74 (b) Twitter_407, τ=0.84 (c) out_arenas, τ=0.74

(d) Brightkite, τ=0.68 (e) DBLP, τ=0.68 (f) ACM, τ=0.72

Figure 20 Scalability

in Table 7) for the advanced index construction and compare it to the baseline index. We see from Table 7 that as expected the time to create the KD-Index is more than CD-Index. The time cost of the KD-Index varies from at least 2-5 times over the time cost of creating CD-Index.

Space cost of index construction We evaluate the space cost to create the indices for different datasets. We also vary the number of diversity values (represented by $|\mathbb{T}|$ in Table 7) for the advanced index construction and compare it to the baseline index. We see from Table 7 that the space cost of KD-Index is more than CD-Index. The space cost of the KD-Index varies from at least 2 times over the space cost of CD-Index. There is significant decrease in the index space cost in comparison to the input size graph dataset as presented in Table 6 because we are only making the indices by setting the diversity threshold at 0.5 and not at 0. This is because the user is interested in diversity values greater than at least 0.5. Thus, the majority of the nodes are not diversed enough in terms of their attributes to be saved in the index. This reduces the size significantly as shown by the experiments.

Table 7 Space and time cost for indices created for real world datasets

Dataset	Space cost (in MB)		Time cost (in secs)	
	$|\mathbb{T}| = 1$	$|\mathbb{T}| = 5$	$|\mathbb{T}| = 1$	$|\mathbb{T}| = 5$
Facebook_0(Fb_0)	0.143	0.154	0.006	0.016
Twitter_407	0.152	0.160	0.008	0.024
out_arenas	0.72	0.79	0.027	0.027
Brightkite	7.31	7.61	0.400	1.526
DBLP	40.73	41.05	2.555	7.381
ACM	41.28	44.49	4.578	9.92

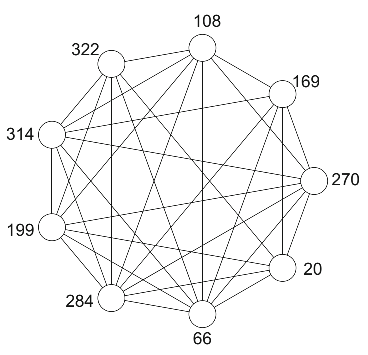
6.3.2 Effectiveness evaluation

We use the Facebook_0 dataset for the motivating example, where a user is associated with a language that he/she can speak/understand. Let us consider the scenario from Figure 6, where we want to promote a product across different Facebook pages in different languages. In such a case, we need to find a set of users whose attributes are diversified in terms of the number of languages shared by them. We also need the users to be socially connected as this makes the marketing in different languages coherent and the users can relay their ideas more clearly. The community that we retrieve is a diversified community where for a certain level of diversification we achieve the most socially cohesive diversified group of users. The level of diversification acts as a diversity threshold.

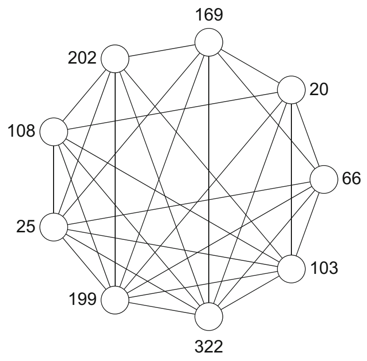
Following this, we conduct a case study on Facebook_0 dataset. We delete the nodes with no languages attributes to maintain consistency across. In comparison to a k-core model which only considers social constraints, our model finds a community which is diversified in terms of users attributes but also strongly connected socially, i.e., the most cohesive. Next, for diversity threshold τ=0.5, the community is found for maximal k=6, as shown in Figure 21a. The community found 10 users and the total languages shared among themselves is 10 out of the 14 languages found in the dataset. As we increase τ to 0.6, 0.7, 0.8 we get the values of k to be 5, 4, 2 respectively and the total languages shared to be 10, 9, 8 respectively. Also, the number of users decreases to 8, 5, 4 respectively. We can see that in the most diverse case, i.e., when τ=0.80, the dataset can get a community with at most 8 languages shared across the 4 users of the community.

So we can deduce that all the users within a community share a set of distinct language set which is guaranteed by the attribute diversity threshold. On the other hand, the users are also socially connected directly or through their friends, which makes marketing a product across different languages more coherent as specified in the motivating example.

We tweak the Facebook_0 dataset by assigning a single common language to all the users to maintain consistency. We found that the maximum τ existed for 0.74 with optimal k=2 covering 9 languages out of the total 15 (14 + 1 added to every node). Whereas when τ=0.5, 0.6, 0.7, we found k = 6, 5 and 2 respectively with attributes shared being 11, 10 and 9 respectively. This also shows when users speak at least one common language then the



(a) Maximal k=6



(b) Maximal k=6 with 1 random variable

Figure 21 Result for Facebook_0 when τ=0.5

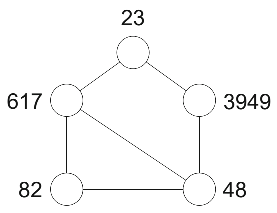


Figure 22 Case study on IMDb with maximal k=2 when τ=0.75

maximum language diversity is moderately found for τ=0.74. For diversity threshold τ=0.5, the community is found for maximal k=6, as shown in Figure 21b.

We conducted another case study on the IMDb dataset with the same settings as specified in Section 6.1.2.

The community found by the ADC-MC model contains 6 film personalities and 7 social relationships between them. The diversity threshold was set at 0.75 to get the maximum social cohesiveness of 2, i.e., each personality is connected to at least 2 other personalities. This model finds a diversified community where each member is diverse with every other member in terms of attributes shared between them, which is quite different as compared to the k-core model. In Figure 22, the result for the ADC-MC model is depicted for IMDb.

7 Related works

Community search in attributed graph In [30, 31], Li et al. proposed a skyline community model for searching communities in attributed graph. Zhang et al. proposed (k, r)-core community model that considers k-core and pairwise vertices similarity [49, 50]. Fang et. al. [18], proposed a community model that is sensitive to query attributes. Liu et al. [35] and Zhang et al. [52], both found a community maximising different attribute score and are structurally cohesive. Zhu et al. [55] proposed a community model that ensures that both structure and attribute cohesiveness are guaranteed for a user given query vertex sharing sharing maximum keyword set. Li et al. [10] proposed a parameter-free contextual community model for attributed community search. In [2, 19, 21, 42] different community models considering spatial closeness were studied. Al-Baghdadi et al. [2] proposed a community model with high social influence, spatially close, and covering certain keywords. Lou et al. [36] proposed a model for attribute constrained co-located community search. In [46] the influence was calculated over propagation probability of an edge above a threshold. Li et al. [28] investigated the problem of influence spanning maximization in location-aware social networks. Li et al. [27] proposed a model for personalised influential topic search that finds influential topics related to a user. Li et al. [27] proposed a model for community diversified influence maximisation, where diversity is the distinct number of communities influenced. In [40] community search over multiple query nodes was proposed. Wang et al. [43] proposed a community model where vertices are closer to a query vertices based on a closeness score. Recently, Lu et al. [12] proposed a diversified geo-social community with attribute diversification. Anwar et al. [3] discovered and tracked time-sensitive activity in dynamic social networks for a user given query topics. Community models considering influence were studied in [5, 29, 32, 46]. In [5, 13, 32], they use max-min objective function as well. However, they find that the influential community and the score are defined on vertices. In our work, we study attribute diversified community problem and the score is defined on relationship for the first 2 models and between every pairwise vertices in the last model.

Community detection in attributed graph Works including [37] considered graph structure with LDA model to detect attributed communities. Unified distance [54] was also considered for detecting attributed communities. In [54], attributed communities were detected by using proposed structural/attribute clustering methods, in which structural distance was unified by attribute weighted edges. Xu et al. [47] proposed a Bayesian based model. Ruan et al. [38] proposed an attributed community detection method that links edges and content, filters some edges that were loosely connected from content perspective, and partitions the remaining graphs into attributed communities. In [25], Huang et al. proposed a community model considering attributes based on an entropy-based model. Li et al. [33] proposed a novel embedding approach to search attributed communities by exploiting the inherent community structures through memberships of the underlying community. Recently, Wu et al. proposed an attributed community model [45] based on an attributed refined fitness model. Yang et al. [48] proposed a model using probabilistic generative model. Zhang et al. [51] proposed a model to find structural diversity in edges for an ego-network. Hsu et al. [24] proposed a diversity model for group formation without considering structural cohesiveness. They also used average diversity objective function.

To read more about the different community search models presented till date, one can read [20] survey paper who have conducted a thorough review of existing works. In [22] extensive literature on other community models can be read further.

8 Conclusion

In this paper, we propose attribute diversified community search. By presenting real world application scenarios, different attribute diversified community models are introduced where attribute diversification takes roles of objective and constraint. For each of the community model, the proposed solutions with optimisations for speeding up the search are discussed in extensive detail. We conducted extensive experiments for each problem to show the effectiveness and efficiency of the algorithms proposed.

Acknowledgements The work was supported by Australia Research Council discovery projects DP170104747 and DP200103700.

References

1. Akbas, E., Zhao, P.: Truss-based community search: a truss-equivalence based indexing approach. PVLDB **10**(11), 1298–1309 (2017)
2. Al-Baghdadi, A., Lian, X.: Topic-based community search over spatial-social networks. Proc VLDB Endowment **13**(12), 2104–2117 (2020)
3. Anwar, M.M., Liu, C., Li, J.: Discovering and tracking query oriented active online social groups in dynamic information network. World Wide Web **22**(4), 1819–1854 (2019)
4. Batagelj, V., Zaversnik, M.: An o(m) algorithm for cores decomposition of networks. arXiv:0310049 (2003)
5. Bi, F., Chang, L., Lin, X., Zhang, W.: An optimal and progressive approach to online search of top-k influential communities. PVLDB **11**(9), 1056–1068 (2018)

6. Bron, C., Kerbosch, J.: Algorithm 457: Finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973)
7. Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. In: Annual Symposium on Foundations of Computer Science, pp. 649–658 (2012)
8. Cai, G., Sun, Y.: The minimum augmentation of any graph to a k edge connected graph. Networks **19**(1), 151–172 (1989)
9. Chang, L., Yu, J.X., Qin, L., Lin, X., Liu, C., Liang, W.: Efficiently computing k-edge connected components via graph decomposition. In: SIGMOD, pp. 205–216. ACM (2013)
10. Chen, L., Liu, C., Liao, K., Li, J., Zhou, R.: Contextual community search over large social networks. In: ICDE, pp. 88–99. IEEE (2019)
11. Chen, L., Liu, C., Zhou, R., Li, J., Yang, X., Wang, B.: Maximum co-located community search in large scale social networks. PVLDB **11**(10), 1233–1246 (2018)
12. Chen, L., Liu, C., Zhou, R., Xu, J., Yu, J.X., Li, J.: Finding effective geo-social group for impromptu activities with diverse demands. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 698–708 (2020)
13. Chen, S., Wei, R., Popova, D., Thomo, A.: Efficient computation of importance based communities in web-scale networks using a single machine. In: International on Conference on Information and Knowledge Management, pp. 1553–1562. ACM (2016)
14. Chowdhary, A.A., Liu, C., Chen, L., Zhou, R., Yang, Y.: Finding attribute diversified communities in complex networks. In: DASFAA, vol. 2020 (2020)
15. Cohen, J.: Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report **16** (2008)
16. Eppstein, D., Löffler, M., Strash, D.: Listing all maximal cliques in sparse graphs in near-optimal time. In: International Symposium on Algorithms and Computation, pp. 403–414. Springer (2010)
17. Eppstein, D., Löffler, M., Strash, D.: Listing all maximal cliques in large sparse real-world graphs. J. Exp. Algorithmics (JEA) **18**, 3–1 (2013)
18. Fang, Y., Cheng, R., Chen, Y., Luo, S., Hu, J.: Effective and efficient attributed community search. VLDB J. **26**(6), 803–828 (2017)
19. Fang, Y., Cheng, R., Li, X., Luo, S., Hu, J.: Effective community search over large spatial graphs. PVLDB **10**(6), 709–720 (2017)
20. Fang, Y., Huang, X., Qin, L., Zhang, Y., Zhang, W., Cheng, R., Lin, X.: A survey of community search over big graphs. The VLDB Journal (2019)
21. Fang, Y., Wang, Z., Cheng, R., Wang, H., Hu, J.: Effective and efficient community search over large directed graphs. TKDE (2018)
22. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3), 75–174 (2010)
23. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989)
24. Hsu, B.Y., Shen, C.Y.: On extracting social-aware diversity-optimized groups in social networks. In: 2018 IEEE Global Communications Conference (GLOBECOM), pp. 206–212. IEEE (2018)
25. Huang, X., Cheng, H., Yu, J.X.: Dense community detection in multi-valued attributed networks. Inform. Sci. **314**(C), 77–99 (2015)
26. Koch, I.: Enumerating all connected maximal common subgraphs in two graphs. Theor. Comput. Sci. **250**(1-2), 1–30 (2001)
27. Li, J., Cai, T., Deng, K., Wang, X., Sellis, T., Xia, F.: Community-diversified influence maximization in social networks. Inf. Syst. **92**, 101522 (2020)
28. Li, J., Sellis, T., Culpepper, J.S., He, Z., Liu, C., Wang, J.: Geo-social influence spanning maximization. IEEE Trans. Knowl. Data Eng. **29**(8), 1653–1666 (2017)
29. Li, J., Wang, X., Deng, K., Yang, X., Sellis, T., Yu, J.X.: Most influential community search over large social networks. In: ICDE, pp. 871–882. IEEE (2017)
30. Li, R., Qin, L., Ye, F., Wang, G., Yu, J.X., Xiao, X., Xiao, N., Zheng, Z.: Finding skyline communities in multi-valued networks. VLDB J. **29**, 1407–1432 (2020)
31. Li, R.H., Qin, L., Ye, F., Yu, J.X., Xiao, X., Xiao, N., Zheng, Z.: Skyline community search in multi-valued networks. In: SIGMOD, pp. 457–472. ACM (2018)
32. Li, R.H., Qin, L., Yu, J.X., Mao, R.: Influential community search in large networks. PVLDB **8**(5), 509–520 (2015)
33. Li, Y., Sha, C., Huang, X., Zhang, Y.: Community detection in attributed graphs: an embedding approach. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
34. Liu, C., Chen, L., Zhou, R., Chowdhary, A.A.: Attribute diversified community search. In: Qin, L., Zhang, W., Zhang, Y., Peng, Y., Kato, H., Wang, W., Xiao, C. (eds.) Software Foundations for Data

Interoperability and Large Scale Graph Data Analytics, pp. 3–17. Springer International Publishing, Cham (2020)

35. Liu, Q., Zhu, Y., Zhao, M., Huang, X., Xu, J., Gao, Y.: Vac: Vertex-centric attributed community search. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), vol. 937–948. IEEE (2020)
36. Luo, J., Cao, X., Xie, X., Qu, Q., Xu, Z., Jensen, C.S.: Efficient attribute-constrained co-located community search. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1201–1212. IEEE (2020)
37. Nallapati, R.M., Ahmed, A., Xing, E.P., Cohen, W.W.: Joint latent topic models for text and citations. In: SIGKDD, pp. 542–550. ACM (2008)
38. Ruan, Y., Fuhry, D., Parthasarathy, S.: Efficient community detection in large networks using content and links. In: WWW, pp. 1089–1098. ACM (2013)
39. Seidman, S.B.: Network structure and minimum degree. Soc. Netw. **5**(3), 269–287 (1983)
40. Sun, H., Huang, R., Jia, X., He, L., Sun, M., Wang, P., Sun, Z., Huang, J.: Community search for multiple nodes on attribute graphs. Knowl. Based Syst. **193**, 105393 (2020)
41. Wang, H.C., Fussell, S.R., Cosley, D.: From diversity to creativity: Stimulating group brainstorming with cultural differences and conversationally-retrieved pictures. In: Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, pp. 265–274 (2011)
42. Wang, K., Cao, X., Lin, X., Zhang, W., Qin, L.: Efficient computing of radius-bounded k-cores. In: ICDE, pp. 233–244. IEEE (2018)
43. Wang, Z., Wang, W., Wang, C., Gu, X., Li, B., Meng, D.: Community focusing: yet another query-dependent community detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 329–337 (2019)
44. Wen, D., Qin, L., Zhang, Y., Chang, L., Chen, L.: Enumerating k-vertex connected components in large graphs. In: ICDE, pp. 52–63. IEEE (2019)
45. Wu, P., Pan, L.: Mining application-aware community organization with expanded feature subspaces from concerned attributes in social networks. Knowl.Based Syst. **139**, 1–12 (2018)
46. Xu, J., Fu, X., Wu, Y., Luo, M., Xu, M., Zheng, N.: Personalized top-n influential community search over large social networks. World Wide Web **23**(3), 2153–2184 (2020)
47. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: A model-based approach to attributed graph clustering. In: SIGMOD, pp. 505–516. ACM (2012)
48. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: ICDM, pp. 1151–1156. IEEE (2013)
49. Zhang, F., Lin, X., Zhang, Y., Qin, L., Zhang, W.: Efficient community discovery with user engagement and similarity. The VLDB Journal **28**(6), 987–1012 (2019)
50. Zhang, F., Zhang, Y., Qin, L., Zhang, W., Lin, X.: When engagement meets similarity: efficient (k, r)-core computation on social networks. PVLDB **10**(10), 998–1009 (2017)
51. Zhang, Q., Li, R.H., Yang, Q., Wang, G., Qin, L.: Efficient top-k edge structural diversity search. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 205–216. IEEE (2020)
52. Zhang, Z., Huang, X., Xu, J., Choi, B., Shang, Z.: Keyword-centric community search. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 422–433. IEEE (2019)
53. Zhou, R., Liu, C., Yu, J.X., Liang, W., Chen, B., Li, J.: Finding maximal k-edge-connected subgraphs from a large graph. In: EDBT, pp. 480–491. ACM (2012)
54. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB **2**(1), 718–729 (2009)
55. Zhu, Y., He, J., Ye, J., Qin, L., Huang, X., Yu, J.X.: When structure meets keywords: Cohesive attributed community search. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1913–1922 (2020)