Attribute Diversified Community Search

Chengfei Liu(✉), Lu Chen, Rui Zhou, and Afzal Azeem Chowdhary

Swinburne University of Technology, Melbourne, Australia
{cliu,luchen,rzhou,achowdhary}@swin.edu.au

Abstract. 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. In recent year, community search in attributed graphs has begun to attract attention, which aims to find communities that are both structure and attribute cohesive. Whereas, searching a community that is structure cohesive but attribute diversified, denoted as attribute diversified community search, is still at preliminary 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 objective, query requirement, and constraint. We also discuss major techniques for speeding up the attribute diversified community search.

1 Introduction

Graphs have emerged as a powerful model for representing different types of data, such as social networks and collaboration networks. 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. Nevertheless, most of the previous studies [1,4,5,10,24,29] 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, community search in graphs having attributes called attributed graphs has begun to attract attention [6,7,11,15,20,28]. These works endeavour to find communities that are both structure and attribute cohesive. Besides, there are also a few works [18] that aim to find communities which are attribute diversified among them. However, a study for community search that takes serious consideration of structure cohesiveness but attribute diversification within a community is still at preliminary stage.

In this paper we focus on introducing our recent works for attribute diversified community search, including three attribute diversified community models in which the attribute diversification takes different roles for presenting an objective, a query requirement, and a constraint.

L. Qin et al. (Eds.): SFDI 2020/LSGDA 2020, CCIS 1281, pp. 3–17, 2020.
https://doi.org/10.1007/978-3-030-61133-0_1

Maximizing Attribute Diversification. Discovering a community with members as diversified as possible has numerous applications. One example 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. Other examples are: gathering socially connected experts of different marketing fields to brainstorm a marketing session of different new products, selecting a panel of concerted engineers with different technological expertise for reviewing and testing different products to show the collective information pool of the panel, etc. For these applications, since they target community members for innovations and there are evidences that maximizing diversity leads to creativity [22], the desired community would be preferred to maximise the attribute diversity of its members [9].

Attribute Diversification with Specific Requirement. For some applications, the diversification requirements could be specific. Let us consider a real event happened in 2019. A small town in Australia was devastated by the severe bushfire, which results in at least 11 damaged properties and 33 people injured. The town needs community spirit to rebuild. This naturally arises the needs of several activities with diverse demands. A group needs to be formed urgently to react on the disaster, with at least 3 members having expertise in building temporary accommodations, 5 doctors, 4 psychologists, 2 members having the expertise in community support, etc. Each member may contribute to as many skills as possible in this kind of group. Due to damaged properties, a construction team also needs to be built for rebuilding these properties, with members having different skills, such as at least 2 architects, 11 members handling masonry, 5 members dealing with welding, etc. Due to the intensive labouring, each member may contribute at most 2 skills in this kind of construction teams since multi-tasking may lead to multi-failing. Due to the disaster, people may suffer a lot mentally. To help relief psychological pressure from these people, it would be great to organize an improvised music show to soothe them, which needs to discover musicians to form a band. The found musicians may be able to play multiple instruments. However, since they perform as a band, each of them shall focus on a single instrument. From these examples, it is clear that, apart from social cohesiveness and spatial closeness requirements, an effective community model for impromptu activities with diverse demands should allow people to express specific diversification requirements including: 1) collective capabilities of the group w.r.t. a particular skill, e.g., at least 3 members have expertise in building temporary accommodation; and 2) capacity of each member on maximum contribution the member can make, e.g., at most 2 skills in a construction team. This motivates us to study how to find an attribute diversified geo-social group with specific diversification requirements [8].

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. For instance, assume that we need to find a group of organisers for organising a conference. To make the organisation smooth, the organisers are expected to communicate and collaborate with each other extensively. The more that organisers identify with each other, the more likely they are to believe that they hold similar goals for successfully organising the conference. On the other hand, to make the conference accept various ideas, we also expect that the organisers would jointly share a variety of domains. Similar applications include promoting a product through commonly associated experts of difference domains, team formations for maximising productivity, etc. Motivated by these applications, we introduce a novel community model that considers attribute diversification as a constraint while maximising the structure cohesiveness as the primary searching objective.

Road Map. The rest of this paper is organized as follows. In Sect. 2, we introduce and discuss basics for attributed graphs. In Sects. 3, 4, and 5 we discuss our recent attribute diversification community search works. We discuss the related works and conclude this paper in Sects. 6 and 7.

2 Preliminaries

In this section, we first formally introduce the commonly used community cohesiveness metrics and attributes diversification metrics.

An attributed graph is denoted as $G = (V, E, A)$, where $V(G)$, $E(G)$, A denote the set of vertices in G, the set of edges in G, and the set of attributes 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. A triangle in G is a cycle of length 3. A triangle induced on vertices $u, v, w \in V(G)$ is denoted as $\triangle_{uvw}$ and when these vertices are not specified we omit the subscript. Given a subgraph $H \subseteq G$, $Tri(H)$ denotes the set of triangles in H.

2.1 Social Cohesiveness Metrics

Coreness. Coreness is defined according to the degree of every vertex.

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.*

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 vertices. A k-core is maximal if it cannot be extended.

Trussness. Trussness is defined based on the number of triangles each edge is involved in a graph. In general, given a subgraph $H \subseteq G$, we use $\triangle_{uvw}$ to denote a triangle, a cycle with length of 3, consisting of vertices $u, v, w \in V(H)$.

Support. The support of an edge $e(u, v) \in E(H)$, denoted by $sup(e, H)$, is the number of triangles containing e, i.e., $sup(e, H) = |\{\triangle_{uvw} : w \in N(v, H) \cap N(u, H)\}|$, where $N(v, H)$ and $N(u, H)$ are the neighbours of u, v in H correspondingly.

Minimum Subgraph Trussness. The trussness for a subgraph H is defined as an integer k that is 2 plus the minimum possible support for edges in $E(H)$. That is, the minimum subgraph trussness defines that for every edge $e \in E(H)$, the number of triangles in which e participates shall be no less than k - 2.

Definition 2. c-truss constraint. *A subgraph H satisfies c-truss constraint if the trussness of H is c, and c is connected.*

Intuitively, if H satisfies c-truss constraint, the vertices of an edge in H have at least c-2 common neighbours in H, every vertex in H has no less than c-1 neighbours and at least c-1 edges have to be deleted in order to make H disconnected. The communication cost of H is at most $\lfloor \frac{2|V(H)|-2}{c} \rfloor$. A H with a large value c indicates strong internal social relationships over vertices.

2.2 Attribute Diversification Metrics

Diversity for Two Vertices. Given a pair of vertices, $u, v \in V(G)$ with attributes $A(u)$ and $A(v)$, a diversity function is defined as $div((u, v)) = 1 - \frac{|A(u) \cap A(c)|}{|A(u) \cup A(v)|}$.

Average Based Diversity. Given H, the attribute diversification of H is measured by $avgDiv(H) = \frac{\sum_{(u,v) \in H} div((u,v))}{|V(H)|}$.

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

3 Discovering a Community Maximizing Attribute Diversity

In this section, we introduce an attribute diversified community search work [9] that aims to find a community maximizing the attribute diversity. We first introduce the community model and search problem, then discuss the search framework and optimizations, respectively.

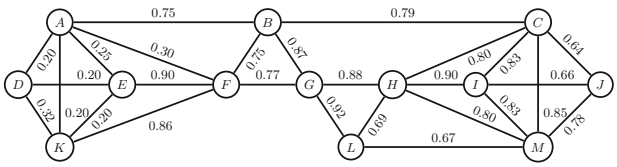


Fig. 1. Graph with edge diversity

Algorithm 1: basicADC(H)

```
1  H* ← φ;
2  basicEnum (H);
3  return H*;
4  Procedure basicEnum (H)
5      H' ← k-core(H);
6      let H' be the set of connected component in H';
7      foreach h ∈ H' do
8          if avgDiv(h) > avgDiv(H*) then
9              H* ← h;
10     foreach h ∈ H' do
11         foreach v ∈ V(h) do
12             basicEnum (h \ {v});
```

3.1 Problem Definition

Attribute Diversified Community. We propose the attribute diversified community model, using k-core and average based diversification metric.

Definition 3. Attribute diversified community. *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 a k-core subgraph;*
- *Maximizing Average Diversity: for $avgDiv(H)$, H is $argmax_{H'}$ $\{avgDiv(H')|H' \subseteq G\}$;*

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

Research Problem. *Find the subgraph $H \subseteq G$ that maximises $avgDiv(H)$.*

Example 1. To briefly show the results of the above problem, we discuss the example shown in Fig. 1. For the attribute diversified community search problem with $k = 2$, the result is the $\{B, C, F, G, H, I, J, L, M\}$ induced subgraph with diversity of 1.44.

3.2 Search Framework

For ease of understanding, we first show the basic enumeration used in the branch and bound algorithm. Algorithm 1 shows the basic enumeration that derives the optimum result. Initially the input of the algorithm is G. By recursively calling itself, Algorithm 1 tries all possible subgraphs of G if the subgraphs may contain the optimum result and checks if there is a feasible solution in the current recursion. 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.

Search Space Reduction. Algorithm 1 also applies space reduction optimisations based on the 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 1 first reduces the input to the maximal k-core, which would transform the input into a set of maximal connected k-cores. Algorithm 1 only tries combinations in each connected k-core. As such, the search space can be reduced significantly.

3.3 Optimisations

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 so far, 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.

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

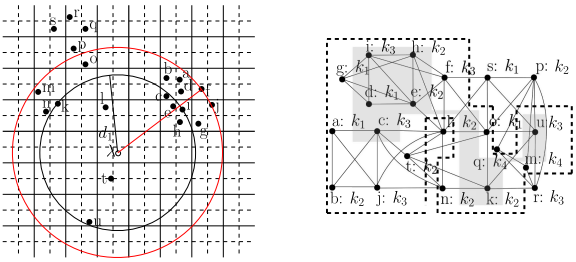
The upper bound based on core property would only be tight when h contains an optimum result with size close to $k+1$. However, it has limited pruning effectiveness when h contains large-size results. Next we study tight bounds for arbitrary h.

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 1. *$ubavg(h)$ is an upper bound for h.*

Approximate Maximum Average Diversity in a Core. The computational cost of $ubavg(h)$ is high. It would take $O(|V(h)|^3)$ if using the algorithm in [13]. However, there is a simple but effective approximate algorithm [3] that can achieve $\frac{1}{2}$-approximation with complexity $O(|E(h)|)$. 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.



(a) Spatial distribution (b) $H_{\leq d_1}$ and $H_{\leq d^*}$

Fig. 2. Running example

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 communities with large average diversity as early as possible. This would make the upper bound based pruning more effective.

4 Discovering Attribute Diversified Geo-Social Group with Specific Requirement

In this section, we tackle the problem of finding an attribute diversified geo-social group with the given specific diversification requirements [8]. We first introduce the query, model and search problem. Then the novel search framework is discussed. After that, the optimizations for speeding up the search are introduced.

4.1 Problem Formulation

Data. We consider an undirected graph data $G = (V, E)$ with network structure, spatial attribute and textual attributes. For each vertex $v \in V(G)$, v has a piece of location information expressed as latitude and longitude denoted as $(v.x, v.y)$, and has a set of keyword attributes denoted as $v.A$.

Since our proposed group model would satisfy minimum keyword, capacity and social constraints while optimizing spatial closeness, we name the proposed model as MKCSSG. Our proposed geo-social model is introduced as follows.

We formally define the query for searching MKCSSG.

Query for MKCSSG. The query Q for MKCSSG consists of a social parameter c (an integer), a set of keywords φ, keyword parameters P (a set of integers), r (an integer), and a location λ (latitude and longitude).

Minimum Keyword and Capacity Constraints. Given a set of query keywords $\varphi = \{k_1, \ldots, k_{|\varphi|}\}$, $P = \{\rho_1, \ldots, \rho_{|\varphi|}\}$, r, and S, MKCC is defined below.

Definition 4. ***Minimum keyword and capacity constraints,*** **MKCC**. *S satisfies MKCC if there is a $v.A' \subseteq \varphi \cap v.A$ for every $v \in V(S)$ such that:*

- *Capacity constraint: $|v.A'| \leq r$,*
- *Minimum keyword constraint (MKC): $\forall\ k_i \in \varphi$, $|V(S_{k_i})| \geq \rho_i$, where $V(S_{k_i})$ is the set of vertices such that for each $v \in V(S_{k_i})$, $v.A'$ contains $k_i \in \varphi$.*

Searching Objective. Now, we formalize the spatial closeness for MKCSSG and the research problem.

Spatial Closeness. Given a query location λ, we consider a distance function to measure the closeness between λ and an MKCSSG S as:

Definition 5. ***Spatial closeness***. $dist(\lambda, S) = max\{\|\lambda - v\| | v \in V(S)\}$,

where $\|\lambda - v\|$ denotes Euclidean distance between v and λ.

Definition 6. **(P, c, r, d)-truss**. *Given $Q = \{\lambda, P, \varphi, c, r\}$ and a distance threshold d, a subgraph $S \subseteq G$ is a (P, c, r, d)-truss, if it satisfies all the conditions: 1) S satisfies MKCC, 2) S satisfies c-truss constraint, 3) $dist(\lambda, S) \leq d$.*

Research Problem. MKCSSG search. *Given $Q = \{\lambda,\ P\ \varphi,\ c,\ r\}$ and G, return (P, c, r, d)-truss S^* so that there is no (P, c, r, d')-truss S' with $d' \leq d$.*

Example 2. An example dataset is shown in Fig. 2, where Fig. 2(a) shows locations for vertices of graph data in Fig. 2(b). Let the query be: $Q = \{\lambda,\ P = \{2, 2, 2\},\ \varphi = \{k_1, k_2, k_3\},\ c = 4,\ r = 1\}$. $\{d, e, f, g, h, i\}$ induced subgraph S^* is the optimum result for Q for this dataset. S^* satisfies social constraint, i.e., every edge in $E(S^*)$ involves no less than 2 triangles. S^* satisfies MKCC. That is, it firstly satisfies capacity constraint, i.e., every vertex contributes to at most one keyword in φ, where $d.A' = \{k_1\}$, $e.A' = \{k_2\}$, $f.A' = \{k_3\}$, $g.A' = \{k_1\}$, $h.A' = \{k_2\}$, $i.A' = \{k_3\}$. Then it satisfies MKC, i.e., with the A' for each vertex (those underlined), the keyword vertex frequency for every query keyword is no less than 2. Last but not least, among all groups satisfying the constraints, S^* is the closest one to λ and the most distant vertex (to λ) in S^* is f.

4.2 Search Framework

We firstly introduce some definitions.

Definition 7. **d radius bounded graph**. *Given a query location λ, a subgraph H and a distance threshold d, d radius bounded graph, denoted as $H_{\leq d}$, is the subgraph of H induced by vertices of H with distance to λ no greater than d.*

We would like to highlight an instance of d radius bounded graph, d^* radius bounded graph, $H_{\leq d^*}$. $H_{\leq d^*}$ has the property below. There is no $H_{\leq d'}$ such that $H_{\leq d'}$ contains MKCSSG and $d' < d^*$.

Optimum Search Space. We refer $H_{\leq d^*}$ as optimum search space since it is just large enough to contain MKCSSG for the query.

The Framework. MKCSSG search framework consists of two stages: expanding stage and reducing stage. During the expanding stage, it intends to quickly identify $H_{\leq d}$ that is just sufficiently large to contain the optimum search space $H_{\leq d^*}$ by exploring $H_{\leq d}$ that progressively gets larger, in which it determines the existence of a subgraph satisfying all constraints. For the reducing stage, to get the optimum result, it attempts to progressively remove the vertex that is the most distant to λ in S^*. The last survived (ρ, c)-truss during the vertices removing process is the optimum result.

4.3 Optimizations for Expanding Stage

Expanding Strategy. We first define an expanding invariant as follows.

Definition 8. *Δ size invariant. Let $\{d_1, d_2, \ldots, d_i\}$ be the series of radius for defining d radius graphs, for any two consecutive d, d′, we define Δ invariant as $\Delta = \frac{|E(H_{\leq d'})|}{|E(H_{\leq d})|}$, in which $\Delta > 1$ must hold.*

The Strategy. The strategy applied for the expanding stage is to maintain Δ size invariant over any two consecutively evaluated $H_{\leq d}$, $H_{\leq d'}$. Applying Δ invariant for expanding stage guarantees two nice properties below. Now, let us show the tight bound that is guaranteed by applying the proposed expanding strategy.

Initial Expanding Range. Intuitively, if the initial search range is close to d^*, the total amount of subgraphs that has to be evaluated to approaching $H_{\leq d^*}$ is less. This motivates us to study a lower bound of d radius subgraph.

Definition 9. *$H_{\leq \underline{d}}$. A subgraph $H_{\leq \underline{d}}$ of H is a lower bound d radius subgraph of $H_{\leq d^*}$ if it satisfies conditions: 1) $H_{\leq \underline{d}}$ is connected, 2) $H_{\leq \underline{d}}$ satisfies minimum keyword constraint and 3) there is no $H' \subseteq H_{\leq \underline{d}}$ such that H' satisfies the first two constraints and $dist(\lambda, H') < dist(\lambda, H_{\leq \underline{d}})$.*

Checking (ρ, c)-truss in d Radius Subgraph. To simplify the discussion, for any two consecutive $H_{\leq d}$ and $H_{\leq d'}$ with $\frac{|H_{\leq d'}|}{|H_{\leq d}|} = \Delta$, let us introduce a new notation $H_{d' \setminus d}$ to denote the subgraph of $H_{\leq d'}$ induced by vertices appearing in edges of $E(H_{\leq d'}) \setminus E(H_{\leq d})$.

We propose two techniques to speed up (ρ, c)-truss checking below.

Lazy (ρ, c)-truss checking strategy. Given $H_{\leq d}$, we only apply (ρ, c)-truss checking on any subgraph potentially containing (ρ, c)-truss, defined as ρ potential subgraph below.

ρ potential subgraph $P_{\leq d}$. A subgraph $P_{\leq d} \subseteq H_{\leq d}$ is defined as ρ potential subgraph if it is connected, satisfies minimum keyword constraint and is *maximal* within $H_{\leq d}$.

The Strategy. Since a (ρ, c)-truss should reside in $P_{\leq d}$, we propose lazy (ρ, c)-truss checking strategy that applies (ρ, c)-truss constraint checking on every $P_{\leq d}$ in $H_{\leq d}$ only instead of the entire $H_{\leq d}$.

Union with Existing Truss. To avoid graph traversing for checking minimum keyword constraint and connectivity after updating trussness, we propose a solution below. Firstly, we maintain every maximal connected c truss subgraph in every $P_{\leq d}$, each of which is attached with keyword vertex frequency. Secondly, after $P_{\leq d}$ is expanded to $P_{\leq d'}$, we update the maintained c-truss subgraphs if applicable. Although this approach cannot update trussness for existing truss subgraphs precisely, it is sufficient and efficient to check the existence of (ρ, c)-truss in $P_{\leq d'}$. As such, minimum keyword constraint and connectivity checking for truss subgraphs can be performed simultaneously and incrementally.

4.4 Optimizations for Reducing Stage

We will maintain a minimum spanning forest for S (the result of the expanding stage) augmented with aggregated keyword vertex frequency. Notice that initially, every spanning tree in the forest satisfies minimum keyword constraint. After an edge is deleted from S, one of the two cases below may happen.

Case 1: the deleted edge is not in the forest. The remaining subgraphs are still connected and each connected subgraph still satisfies minimum keyword constraint.

Case 2: the deleted edge is in the forest. In this case, one of the tree in the minimum spanning forest is cut into two trees, which may lead to one of the following subcases.

Subcase 1: cannot link the cut trees. We cannot find a replacement edge from the remaining S to link the two trees, which means the subgraph referred by the two trees becomes two disjoint subgraphs. We update keyword vertex frequency for each of the cut tree. After the update, we safely prune the cut tree from the maintained spanning forest if it does not satisfy minimum keyword constraint since they cannot contribute to MKCSSG.

Subcase 2: can link the cut trees. If we can find a replacement edge, the subgraph referred by two cut trees is still connected. We link the two trees with the replacement edge. Keyword vertex frequency remains the same.

To efficiently maintain the above index, we borrow the idea from [14]. Given S, every edge in $E(S)$ is associated with a level progressively increased as edges are deleted, which is equivalent to progressively partitioning S hierarchically. Edges with high level refer to a more restricted part of S. In contrast, edges with low level refer to a more general part of S (super graphs of the high level subgraphs). As such when deleting an edge with a certain level, we do not need to consider any edge with lower level as a replacement edge, which elegantly reduces the search space for finding a replacement edge.

4.5 Optimizations for Keyword Constraint Checking

We will show that MKCC checking for a set of vertices S, query keywords $\varphi = \{k_1, \ldots, k_{|\varphi|}\}$, $P = \{\rho_1, \ldots, \rho_{|\varphi|}\}$ and r can be reduced to an instance of the min cut problem.

The Instance of Maximum Flow Problem. We construct the flow network N based on φ, P, r and S. N consists of different types of nodes below. For each keyword in φ, we create a keyword node. For each vertex in S, we create a vertex node. Additionally, we create a source node s and a sink node t. The edges and capacities for N are as follows. For each vertex node n, we create an edge from s to n with capacity of r. For each keyword node n' representing k_i, we create an edge from n' to t with capacity of ρ_i. In addition, there is an edge between a vertex node n and a keyword node n' (from n to n') if the keyword attributes of n representing vertex contain the query keyword represented by n', and the capacity between n and n' is set to ∞.

Lemma 2. *S satisfies MKCC if there exists a min cut for N whose T part contains the node t only.*

We adopt preflow–push (push-relabel) algorithm to solve the min cut problem. As such, the time complexity of MKCC checking for MKCSSG search is shown below.

MKCC checking complexity for the expanding stage. This part can be bounded by $\mathcal{O}\left((1 + \Delta^3 + \frac{1}{\Delta^3 - 1}) \times |V(H_{\leq d^*})|^3\right)$, assuming $|\varphi| \ll |V(H_{\leq d^*})|$. As discussed previously, by letting $\Delta = 2$, the time complexity becomes the minimum, $\mathcal{O}\left(|V(H_{\leq d^*})|^3\right)$.

MKCC checking complexity for the reducing stage. This part can be bounded by $\mathcal{O}\left(|V(H_{\leq d^*})|^3\right)$ as well, by taking advantage of the preflow-push algorithm.

5 Discovering a Community with Attribute Diversification Constraint

In this section we introduce an attribute diversified community search work that focuses on finding a community maximizing the structure cohesiveness while maintaining the attribute diversity to a certain level. The community model is discussed firstly. Then the solution vision is introduced.

5.1 Problem Formulation

We formally define the *subgraph diversity* below.

Definition 10. Subgraph diversity. *Given a subgraph $H \subseteq G$, we define its subgraph diversity as:*

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

Definition 11. (k,τ)**-core**. *Given a subgraph $H \subseteq G$, a pairwise vertex diversity threshold τ, H is a (k,τ)-core if H satisfies the following constraints simultaneously: 1) H is connected, 2) $\forall v \in V(H)$, $deg(v) \geq k$, 3) $\tau(H) \geq \tau$.*

Research Problem. ***Most cohesive diversified community search***. *Given an attributed graph G, a user given attribute diversity threshold τ, find a (k,τ)-core $H \subseteq G$ such that there is no (k',τ)-core H' with k' greater than k.*

5.2 Solution Vision

We introduce a definition and an observation as follows.

Definition 12. Diversity Subgraph. *Given a subgraph diversity threshold τ, let D denote a new graph named as 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).*

We are ready to discuss the vision of our proposed algorithms.
Baseline. The baseline algorithm enumerates all maximal cliques of D using the state-of-the-art clique enumeration algorithm. When a maximal clique C is generated, it runs core decomposition for $G(C)$. If the largest core number of $G(C)$ is greater than the largest k of the (k,τ)-core found so far, then $G(C)$ contains the best (k,τ)-core found so far. After evaluating all the maximal cliques in D, the optimum result is found. The correctness of the baseline algorithm is clear since the vertices in a most cohesive (k,τ)-core must be resident at one of the maximal cliques in D.

Core Based Heuristic. To speed up the baseline, we consider a core based heuristic: vertices in a large core will be considered prior to the vertices in a small core. Using this heuristic, we may terminate the search quite early, i.e., let k be the largest k for (k,τ)-core found so far, if the core number of the current explored vertex is smaller than k, then we can terminate the search.

Advanced Heuristic. The core based heuristic just 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. We propose a novel index denoted as KD-Index to help us identify the vertices that are promising from both the structure cohesive and the attribute diversified perspectives. To make the index general for different τ, we pre-compute promising vertices for different diversity thresholds. When a query τ is given, we evaluate the precomputed index that is just smaller than τ for speeding up the search.

Better Local Enumeration Order. The advanced heuristic provides an overall search order. However, when evaluating each promising subgraph, the local

search order can be further optimised for speeding up the search. We propose to use the degeneracy order when evaluating vertices in a promising subgraph, which can nicely bound the search depth of the promising subgraph to the degeneracy of the subgraph. The degeneracy of the subgraph is much smaller than the total number of vertices contained in the subgraph.

6 Related Work

Community Search in Attributed Graph. In [18], Li et al. propose a skyline community model for searching communities in attributed graph. Zhang et al. propose (k, r)-core community model that considers k-core and pairwise vertex similarity [28]. Fang et al. propose a community model that is sensitive to query attributes. In [16], an attributed community model is proposed by using k-truss for capturing social cohesiveness and the resultant community shall contain attributes similar with query attributes. In [12,23], community models considering spatial closeness are studied. The works above would find communities with users having similar attributes while our work find communities with users having diversified attributes. In [6], a parameter-free contextual community model is studied. Community models considering influence are studied in [2,17,19]. In [2,19], the authors use max-min objective function, which aims to find influential communities where scores are defined on vertices.

Community Detection in Attributed Graph. Works including [21] consider graph structure with LDA model to detect attributed communities. Unified distance [30] is also considered for detecting attributed communities. In [30], attributed communities are detected by using proposed structural/attribute clustering methods, in which structural distance is unified by attribute weighted edges. Xu et al. [26] propose a Bayesian based model. In [15], Huang et al. propose a community model considering attributes based on an entropy-based model. Recently, Wu et al. propose an attributed community model [25] based on an attributed refined fitness model. Yang et al. [27] propose a model using probabilistic generative model.

7 Conclusion and Open Problems

In this paper, we introduce our recent works on attribute diversified community search. Based on the detailed real application scenarios, different attribute diversified community models are introduced where the attribute diversification takes the roles of objective, query requirement and constraint. For each of the community model, the search framework as well as major optimizations for speeding up the search are discussed in great detail.

Although we have made a fair effort for discovering attribute diversified communities, however the study is still at preliminary stage. We conclude by introducing an open problem for attribute diversified community search.

How to effectively model and efficiently search top-r attribute diversified community considering both inter and intra attribute diversifications?

Acknowledgments. This work is jointly supported by the ARC Discovery Projects under Grant No. DP170104747 and DP200103700.

References

1. Batagelj, V., Zaversnik, M.: An o(m) algorithm for cores decomposition of networks. arXiv preprint cs/0310049 (2003)
2. 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)
3. 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)
4. Cai, G., Sun, Y.: The minimum augmentation of any graph to a k edge connected graph. Networks **19**(1), 151–172 (1989)
5. 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 (2013)
6. Chen, L., Liu, C., Liao, K., Li, J., Zhou, R.: Contextual community search over large social networks. In: ICDE, pp. 88–99. IEEE (2019)
7. 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)
8. Chen, L., Liu, C., Zhou, R., Xu, J., Yu, J.X., Li, J.: Finding effective geosocial group for impromptu activities with diverse demands. In: SIGKDD. ACM (2020)
9. Chowdhary, A.A., Liu, C., Chen, L., Zhou, R., Yang, Y.: Finding attribute diversified communities in complex networks. In: Nah, Y., Cui, B., Lee, S.-W., Yu, J.X., Moon, Y.-S., Whang, S.E. (eds.) DASFAA 2020. LNCS, vol. 12114, pp. 19–35. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59419-0_2
10. Cohen, J.: Trusses: cohesive subgraphs for social network analysis. National Security Agency Technical Report 16 (2008)
11. Fang, Y., Cheng, R., Chen, Y., Luo, S., Hu, J.: Effective and efficient attributed community search. VLDB J. **26**(6), 803–828 (2017). https://doi.org/10.1007/s00778-017-0482-5
12. Fang, Y., Cheng, R., Li, X., Luo, S., Hu, J.: Effective community search over large spatial graphs. PVLDB **10**(6), 709–720 (2017)
13. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989)
14. Holm, J., De Lichtenberg, K., Thorup, M., Thorup, M.: Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. J. ACM **48**(4), 723–760 (2001)
15. Huang, X., Cheng, H., Yu, J.X.: Dense community detection in multi-valued attributed networks. Inf. Sci. **314**(C), 77–99 (2015)
16. Huang, X., Lakshmanan, L.V.: Attribute-driven community search. PVLDB **10**(9), 949–960 (2017)
17. 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)
18. Li, R.H., et al.: Skyline community search in multi-valued networks. In: SIGMOD, pp. 457–472. ACM (2018)
19. Li, R.H., Qin, L., Yu, J.X., Mao, R.: Influential community search in large networks. PVLDB **8**(5), 509–520 (2015)

20. Li, Y., Sha, C., Huang, X., Zhang, Y.: Community detection in attributed graphs: an embedding approach. In: AAAI (2018)
21. 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)
22. Wang, H.C., Fussell, S.R., Cosley, D.: From diversity to creativity: stimulating group brainstorming with cultural differences and conversationally-retrieved pictures. In: CSCW, pp. 265–274. ACM (2011)
23. Wang, K., Cao, X., Lin, X., Zhang, W., Qin, L.: Efficient computing of radius-bounded k-cores. In: ICDE, pp. 233–244. IEEE (2018)
24. 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)
25. 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)
26. 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)
27. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: ICDM, pp. 1151–1156. IEEE (2013)
28. 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)
29. 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)
30. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB **2**(1), 718–729 (2009)