# Efficient Detection of Overlapping Communities Using Asymmetric Triangle Cuts

Mojtaba Rezvani, Weifa Liang, *Senior Member, IEEE*, Chengfei Liu, *Member, IEEE* and Jeffrey Xu Yu

**Abstract**—Real social networks contain many communities, where members within each community are densely connected with each other, while they are sparsely connected with the members outside of the community. Since each member can join multiple communities simultaneously, communities in social networks are usually overlapping with each other. How to efficiently and effectively identify overlapping communities in a large social network becomes a fundamental problem in the big data era. Most existing studies on community finding focused on non-overlapping communities based on several well-known community fitness metrics. However, recent investigations have shown that these fitness metrics may suffer free rider and separation effects where the overlapping region of two communities always belongs to the denser one, rather to both of them. In this paper, we study the overlapping community detection problem in social networks that not only takes the quality of the found overlapping communities but also incorporate both free rider and separation effects on the found communities into consideration. Specifically, in this paper we first propose a novel community fitness metric - triangle based fitness metric, for overlapping community detection that can minimize the free rider and separation effects on found overlapping communities, and show that the problem is NP-hard. We then propose an efficient yet scalable algorithm for the problem that can deliver a feasible solution. We finally validate the effectiveness of the proposed fitness metric and evaluate the performance of the proposed algorithm, through conducting extensive experiments on real-world datasets with over 100 million vertices and edges. Experimental results demonstrate that the proposed algorithm is very promising.

**Index Terms**—Overlapping community detection; fitness metrics for overlapping communities; social networks; community detection algorithms.

## 1 Introduction

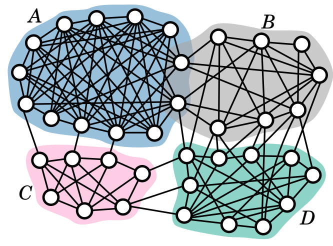
VERTICES in social networks can be clustered into cohesive groups called *communities*, where the vertices within a community are densely connected with each other, while they are sparsely connected to the vertices outside the community. Recent studies [23], [32], [35] have shown that some members of a social network can join multiple communities to broker ideas and access resources from other communities. As a result, communities in social networks are overlapping, rather than exclusive, with each other. The detection of overlapping communities from a social network thus becomes a fundamental problem in big data era, as it has many real applications. For example, for targeted advertisements in a consumer network, detecting overlapping communities helps to identify groups of members with similar shopping preferences, and they will become suitable audiences for an advertisement campaign, as they usually share several shopping preferences [1]. In WWW, web pages with high content commonality can be obtained by detecting overlapping communities of hyperlink networks [6]. In author-collaboration networks, communities reveal research areas and topics that are pursued by different researchers [21]. There are many other applications of overlapping communities, including disease spread controls [25], product recommendations, and mining of structural hole spanners [23], [34].

- M. Rezvani and W. Liang are with the Research School of Computer Science, The Australian National University, Canberra, ACT 2601, Australia. E-mails: mojtaba.rezvani@anu.edu.au, and wliang@cs.anu.edu.au
- C. Liu is with Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia. E-mail: cliu@swin.edu.au
- J. X. Yu is with Department of System Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, China. Email: yu@se.cuhk.edu.hk

The key to identifying high-quality overlapping communities in large-scale networks is an accurate *fitness metric*, which measures the quality of identified communities, in terms of the density of internal edges within a community and sparsity of edges leaving the community. Examples of well-known fitness metrics include Classic Density [24], Relative Density [18], Conductance [14], Subgraph Modularity [17], and Local Modularity [22]. We here assume that all fitness metrics measure the strength of communities. Since some of these fitness metrics (such as conductance) measure the weakness of a community (a smaller fitness value is preferred), we here use the inverse value of these fitness metrics to maximize the strength of communities. Existing fitness metrics for overlapping community detection introduce the following two issues.

One issue with the conductance and local modularity fitness metrics is the *separation effect* that the overlapping region is assigned to only one of the two communities. Fig. 1 shows that the conductance value of community $B - A$ is larger than that of community $B$. Thus, vertices in the overlapping region will be assigned to community $A$ only, as this will result in a larger conductance value. In fact, the vertices in the overlapping region should be assigned to both $A$ and $B$.

Another issue is the presence of the free rider effect [31]. Bandyopadhyay *et al.* [2] proposed an algorithm `FOCS` that



| Fitness metrics | $f(A)$ | $f(B)$ | $f(A \cup B)$ | $f(B - A)$ | $f(A - B)$ | Detected communities | Remarks |
|---|---|---|---|---|---|---|---|
| Classic density (CD) | 4.38 | 2.44 | 3.6 | 2.28 | 4.09 | $A$, $A \cup B$ | Free rider effect |
| Relative density (RD) | 0.85 | 0.47 | 0.84 | 0.48 | 0.73 | $A$, $A \cup B$ | Free rider effect |
| Subgraph modularity (SM) | 5.7 | 0.91 | 5.26 | 0.94 | 2.81 | $A$, $A \cup B$ | Free rider effect |
| Local modularity (LM) | 0.73 | 0.61 | 0.41 | 0.57 | 0.81 | $B$, $A - B$ | Separation effect |
| Conductance (CN) | 0.85 | 0.47 | 0.26 | 0.94 | 0.73 | $A$, $B - A$ | Separation effect |
| Triangle ($\tau(\cdot)$) | 4.11 | 4.875 | 3.454 | 0.545 | 2.85 | $A$, $B$ | Accurate |

Fig. 1. A small network and different fitness values for its communities. While communities $A$ and $B$ share two vertices, fitness metrics CD, RD and SM obtain larger fitness values for communities $A, A \cup B$ (free rider effect), and fitness metrics LM and CN obtain larger values for $B, A - B$ and $A, B - A$, respectively (separation effect).

expands neighborhoods of vertices, using the subgraph modularity fitness metric. Wu *et al.* [31] however showed that the modularity metric and other existing metrics suffer from *free rider effect* [31] or *resolution limit* [9] on found communities, where a community that is always merged with its densest neighboring community will result in a better community in terms of the adopted fitness metric. Fig. 1 illustrates that the values of classic density, relative density and subgraph modularity of community $A \cup B$ are larger than those of community $B$, which means that these fitness metrics can cause free rider effect. We later show that fitness metrics relying on only the internal density usually result in free riders, while fitness metrics relying on only the external sparsity usually cause separation effects.

Despite the importance of both free rider and separation effects on overlapping communities, they have not thoroughly been studied yet. The traditional free rider effect [31] for non-overlapping communities states that the fitness metric value of the resulting community by merging two communities is better than that of either of the communities. This definition, however, may not apply to overlapping communities, due to the fact that the overlapping region of the two communities should belong to both of them.

Although triangle-based approaches entail a strong cohesion among vertices in communities [4], [26], [29], [36], the aforementioned fitness metrics measure the quality of communities based on the most obvious structure in networks - the edges, while ignoring more inherent structures within networks such as triangles. Benson *et al.* [3] and Tsourakakis *et al.* [28] recently extended the conductance metric by introducing motif concepts, and made use of a given motif as the building block to identify communities, where the number of motif instances in a subgraph is the density of the subgraph, and the number of instances that are partially included in the subgraph is its external sparsity. Particularly, these studies show the effectiveness of triangle motifs in the structure of communities. However, the mentioned conductance metric relies heavily on the external sparsity of communities. Under this metric, the overlapping region of two communities is always assigned to the one with more edge connections.

In this paper, we will study the free rider effect on overlapping communities, which has not been studied previously [9], [13], [31]. We will also study the separation effect on overlapping communities. To the best of our knowledge, this is the first overlapping community detection work that considers the quality of overlapping communities, while minimizing both free rider and separation effects on overlapping communities. The main contributions of this paper are as follows.

- We first introduce a new definition of internal density and external sparsity of communities based on 'asymmetric triangle cuts', and propose a new fitness metric for overlapping community detection that mitigates both free rider and separation effects on communities.
- We then formulate a novel overlapping detection problem based on the proposed fitness metric and show its NP-hardness. We instead devise an efficient yet scalable algorithm for the problem.
- We finally conduct experiments to evaluate the performance of the proposed algorithm using real datasets. Experimental results demonstrate that the proposed algorithm outperforms existing methods, in comparison with the ground-truth communities.

The rest of this paper is organized as follows. We first introduce the network model, propose a new fitness metric for overlapping communities, and define the overlapping community detection problem in Section 2. We then show the NP-hardness of the defined problem and devise an algorithm and analyze the time complexity of the algorithm for the problem in Section 3 and Section 4, respectively. We also evaluate the performance of the proposed algorithm in Section 5. We finally review existing methods in Section 6, and conclude in Section 7.

## 2 Preliminaries

In this section, we first introduce the network model and notations. We then propose a new fitness metric - overlapping triangle connectivity metric, for overlapping communities. We finally define the overlapping community detection problem precisely.

### 2.1 Network model

A social network can be modeled as an undirected connected graph $G = (V, E)$, where $V$ is a set of vertices, and $E$ is a set of edges representing the relationships between the vertices. Denote by $N(v)$ the set of neighbors of vertex $v \in V$. The *volume* of a subset of vertices $S$ $(\subseteq V)$ is defined as $vol(S) = \sum_{v \in S} |N(v)|$.

Let $E(S, T)$ be an edge cut between subsets $S$ and $T$ of vertices and $e(S, T)$ the cut size, i.e., $e(S, T) = |E(S, T)|$. Let $e(S)$ represent the number of edges in the induced subgraph $G[S] = (S, E[S])$ of a graph $G(V, E)$ by the vertices in $S$, where $E[S] = \{(u, v) \mid u \in S, v \in S$, and $(u, v) \in E\}$, i.e., $e(S) = |E[S]|$.

A $c$-clique $K_c$ is a complete graph of $c$ vertices. A *triangle* is a cycle of length three. Denote by $\Delta_{uvw}$ the triangle that consists of vertices $u$, $v$ and $w$, and denote by $\Delta_G$ the set of triangles in $G$. The support $sup_G(e)$ of any edge $e = (u, v)$ in $G$ is the number of triangles in which $e$ is contained, which is equal to the number of common neighbors of the two endpoints of edge $e$ ($|N(u) \cap N(v)|$).

Let $\mathscr{C} = \{V_1, V_2, \cdots, V_{|\mathscr{C}|}\}$ be the set of communities in $G$. A triangle $\Delta_{uvw}$ is called a *community triangle* if there is a community $V_i \in \mathscr{C}$ that includes all its vertices; otherwise it is called a *cut triangle*. For a community $V_i$ in $\mathscr{C}$, $\Delta_G(V_i)$ represents the set of triangles formed by the vertices in $V_i$. The *asymmetric triangle cut* $\Delta_G(V_i, V_j)$ between two communities $V_i$ and $V_j$ in $\mathscr{C}$ is defined as the set of cut triangles with each having two vertices in $V_i$ and one vertex in $V_j$. Note that $\Delta_G(V_i, V_j)$ is not necessarily equal to $\Delta_G(V_j, V_i)$.

## 2.2 Community fitness metrics

The vertices in a graph $G(V, E)$ can be allocated to different communities. Let $\mathscr{C} = \{C_1, ..., C_q\}$ be the collection of all communities in $G$, where $C_i \subseteq V$ is a community, $\cup_{i=1}^{q} C_i = V$, and $C_i \cap C_j$ may and may not be empty ($i \neq j$), $1 \leq i, j \leq q$. The *fitness metric* of a community $C \in \mathscr{C}$, denoted by $f(C)$, will determine the degree to which vertices inside $C$ are connected with each other, while separating from the vertices in $V \setminus C$. The fitness metric of a collection of communities $\mathscr{C}$ thus is defined as a summation over the fitness values of all communities in the collection, i.e., $f(\mathscr{C}) = \sum_{C \in \mathscr{C}} f(C)$. In the following we introduce several widely-adopted fitness metrics for community detection [8], [32].

- *Classic density* $\delta(C)$ of a community $C$ [24] is referred to as the average degree of vertices within the community $C$, i.e., $\delta(C) = e(C)/|C|$, where $e(C)$ is the number of edges in the subgraph induced by vertices in $C$.
- *Relative density* $\rho(C)$ of a community $C$ [18] is referred to as the ratio $e(C)$ of the number of edges in community $C$ to the number of edges that have at least one vertex in $C$, i.e., $\rho(C) = e(C)/(e(C) + e(C, V \setminus C))$.
- *Subgraph modularity* $\psi(C)$ of a community $C$ [17], [31] is referred to as the ratio of the number of edges in community $C$ to the number of edges between vertices in $C$ and the vertices in $V \setminus C$, i.e., $\psi(C) = e(C)/e(C, V \setminus C)$. Note that this subgraph modularity [17] is a variant of the traditional modularity [22].
- *Local modularity* $\mu(C)$ of a community $C$ [22] is the ratio of the number of edges in $C$ between boundary vertices and other vertices in $C$ to the number of edges between boundary vertices in $C$ and all other vertices in the network, i.e., $\mu(C) = e(B(C), C)/(B(C), V)$, where $B(C)$ is the boundary set of vertices in $C$, which are adjacent to at least one vertex outside $C$.
- *Conductance* $\sigma(C)$ of a community $C$ [14] is referred to as the ratio of the size of the edge cut to the minimum of the number of edges that have at least one endpoint in $C$ and number of edges that have at least one endpoint in $V \setminus C$, i.e., $\sigma(C) = e(C, V \setminus C)/\min\{vol(C), vol(V \setminus C)\}$. Unlike other fitness metrics, smaller values of conductance are preferred for a community. Therefore, we consider the inverse of this value, and throughout this paper we refer to conductance as $\sigma'(C) = \min\{vol(C), vol(V \setminus C)\}/e(C, V \setminus C)$.

## 2.3 Overlapping community fitness metrics

The aforementioned community fitness metrics are appropriate for non-overlapping community detection. However, they may fail to detect overlapping communities as they may cause *free rider effect* [31] and *separation effect* on the found communities. In other words, these metrics dismiss the overlapping region between two communities by either assigning it to only one of them (separation effect) or merging them into a single community (free rider effect). For example, if we adopt the conductance metric in the social network in Fig. 1, it can only detect community $B - (A \cap B)$ but exclude $A \cap B$ that is densely connected to $B$. This implies that the fitness value $\sigma'(A) + \sigma'(B - (A \cap B))$ is no less than that of $\sigma'(A) + \sigma'(B)$, using the conductance metric. This will result in the separation effect. In the following, we define free rider and separation effects on overlapping communities formally.

***Definition 1 (Separation Effect).*** Given a social network $G = (V, E)$ and a fitness metric $f(\cdot)$, the separation effect happens when for any two communities $A$ and $B$,

$$f(A) + f(B) < \max\{f(A) + f(B - A \cap B), f(B) + f(A - A \cap B)\}. \quad (1)$$

It is noticed that community fitness metrics that rely on the number of edges between communities tend to assign the overlapping region of two communities to only one of them, i.e., to the one with more edges connected. Examples of such fitness metrics include conductance and local modularity metrics in Fig. 1. On the other hand, the primary cause of free rider effects of existing fitness metrics is that the quality of a community is measured in terms of the density of its edges, which can be averaged when merging two communities. That is why after merging community $B$ with the denser community $A$ in Fig. 1, the fitness value of community $A \cup B$ becomes larger than that of community $B$. However, we observe that the fitness value of the merged community $A \cup B$ is not necessarily greater than that of the dense community $A$ since they are weakly connected to each other except their overlapping region. We use this intuition to extend the free rider effect on overlapping communities as follows.

***Definition 2 (Free Rider Effect).*** Given a social network $G = (V, E)$ and a fitness metric $f(\cdot)$, the free rider effect happens when there are two communities $A$ and $B$,

$$f(A) \geq f(A \cup B) \quad \wedge \quad f(B) < f(A \cup B). \quad (2)$$

This definition of the free rider effect implies that if a community $B$ is merged with $A$, they will form a new community $A \cup B$ with a larger fitness value. However, if the fitness value of the resulting community is larger than only one of them, the free rider will happen. Otherwise, the merge will become valid, and such a merge will not incur the free rider effect. Fig. 1 illustrates that fitness metrics such as classic density and relative density can identify community $A$ successfully, but they fail to identify community $B$. They identify community $A \cup B$, instead of $B$, which means that they cause free rider effect.

### 2.4 A new fitness metric based on triangle cuts for overlapping community detection

We here propose a new fitness metric, *overlapping triangle connectivity*, for overlapping community detection that can minimize free rider and separation effects on overlapping communities.

Recent studies [4], [12], [13] have shown that triangles can guarantee a strong cohesion among vertices in a community, by which the vertices in the same community are close to each other (the diameter of a subgraph induced by a community is small [13]), and the connectivity among the vertices in the community is robust (vertices in a community exhibit a strong edge-connectivity [4], [13]). Therefore, a good community fitness metric should favor a large number of triangles within a community. Inversely, the connectivity between communities is not a preferred property, since a large number of edges between two communities may not represent strong connections of vertices between the two communities. We thus make use of the number of triangles, instead of the number of edges, between different communities as a proper fitness metric to measure the connectivity among the communities.

Let $\mathscr{C} = \{C_1, ..., C_q\}$ be the collection of overlapping communities in $G(V, E)$. The *overlapping triangle connectivity* of a community $C \in \mathscr{C}$ is the degree to which the vertices in $C$ are connected with each other and sparsely connected to the rest of vertices in $G$. Thus, all triangles in $G$ can be categorized into two types: *community triangles* and *cut triangles*, where a *community triangle* is a triangle that has all its three vertices in the community, while a *cut triangle* is a triangle that at least one of its three vertices does not lie in the same community as the other two vertices. Since the number of community triangles indicates the strength of cohesion among the vertices in a community, the number of community triangles will be put in the numerator while the number of asymmetric cut triangles will be put in the denominator of the fitness metric. To balance the overlapping region between communities and avoid the free rider effect, the overlapping size of a community with other communities will be put in the denominator of the fitness metric. Finally, the number of vertices contained in a community will be put in the denominator of the fitness metric to avoid over-sized communities and balance the fitness values of communities. We thus define the overlapping triangle connectivity $\tau(C)$, as the ratio of the number of community triangles in $C$ to the sum of the number of cut triangles, the number of vertices in $C$, and the size of overlapping region with the other communities, i.e.,

$$\begin{aligned}\tau(C) \;=\; & |\Delta_G(C)|/(|\{\Delta_{uvw} \,:\, u,v \in C, w \in V \setminus C \;\& \\ & \nexists_{C' \in \mathscr{C}} \; s.t. \; u,v,w \in C'\}| + \sum_{C' \in \mathscr{C}} |C \cap C'| + |C|). \quad (3)\end{aligned}$$

Notice that the number of vertices is applied in the denominator of the fitness metric in Eq.(3) to normalize the value and the ratio of the number of triangles to the number of vertices. Without the term of the number of vertices in the denominator, a larger community would have a larger fitness value. It can be seen from Fig. 1 that the merge of two communities $A$ and $B$ does not increase the fitness value of the resulting community (since the number of vertices increases and the number of triangles per vertex does not increase), using the proposed overlapping triangle connectivity fitness metric $\tau(\cdot)$. Fig. 1 also shows that the fitness value of the resulting community does not necessarily increase by merging a community $B$ to another denser community $A$. Thus, the fitness metric $\tau(\cdot)$ does not result in free rider effect on overlapping communities in this example.

A fitness metric $f(\cdot)$ is said to be *monotonically increasing* if for any two subsets $V_1 \subseteq V$ and $V_2 \subseteq V \setminus V_1$, $f(V_1 \cup V_2) \geq f(V_1)$ always holds. Similarly, $f(\cdot)$ is said to be *monotonically decreasing* if $f(V_1 \cup V_2) \leq f(V_1)$. A fitness function is *non-monotonic* if it neither monotonically increases nor monotonically decreases. Monotonicity of a fitness metric $f(\cdot)$ has several implications such as the occurrence of free rider effect [31] and the existence of an approximation algorithm for community detection under the fitness metric using hill climbing algorithms [19]. In the following we show the defined fitness metric $\tau(\cdot)$ is non-monotonic.

***Lemma 1.*** The defined fitness metric function $\tau(\cdot)$ is a non-monotonic function.

*Proof:* We show the non-monotonicity of function $\tau(\cdot)$, by proving that for a given community $C \in \mathscr{C}$, there exist vertices $v, u \notin C$ such that $\tau(C) \geq \tau(C \cup \{v\})$ while $\tau(C) \leq \tau(\{C \cup \{u\}\})$ as follows.

Let $C$ be a community that consists of a clique $K_{n-1}$ ($n > 4$) and there are two vertices $v \in V \setminus C$ and $u \in V \setminus C$. We first show that there is a vertex $v \in V \setminus C$ such that $\tau(C) \geq \tau(C \cup \{v\})$. The fitness value of clique $K_{n-1}$ is $\tau(K_{n-1}) = \frac{(n-1)(n-2)(n-3)}{6(n-1)}$. Assume that $v$ is connected to some vertices in $C$ but does not form any triangles with the vertices. Now, if vertex $v$ is added to $C$, the fitness value of the resulting community $C \cup \{v\}$ will be $\tau(K_{n-1} \cup \{v\}) = \frac{(n-1)(n-2)(n-3)}{6(n-1+1)}$. Clearly, $\tau(K_{n-1}) > \tau(K_{n-1} \cup \{v\})$, or, $\tau(C) > \tau(C \cup \{v\})$.

We then prove that there is another vertex $u \in V \setminus C$ such that $\tau(C) \leq \tau(C \cup \{u\})$. Let $u$ be a vertex in clique $K_{n-1}$. Removing $u$ and its incident edges from $K_{n-1}$ leaves us with a clique $K_{n-2}$ with the fitness value $\tau(K_{n-2}) = \frac{(n-2)(n-3)(n-4)}{6(n-2)}$. Now, if $u$ is added to $C$, then $\tau(K_{n-2} \cup \{u\}) = \frac{(n-1)(n-2)(n-3)}{6(n-2+1)}$. Since $n > 4$, $\tau(K_{n-2}) < \tau(K_{n-2} \cup \{u\})$. Thus, the fitness value increases. □

Lemma 1 implies that devising an efficient algorithm for overlapping community detection in $G$ with an objective to maximize $\tau(\mathscr{C})$ ($= \sum_{C \in \mathscr{C}} \tau(C)$) is extremely difficult, due to the non-monotonicity of function $\tau(\cdot)$. Instead, we will develop an efficient heuristic algorithm for the overlapping community detection problem.

### 2.5 Problem Definition

Given a social network $G = (V, E)$ and the overlapping triangle connectivity fitness metric $\tau(\cdot)$, *the overlapping community detection problem* in $G$ is to find a collection of overlapping communities $\mathscr{C} = \{C_1, ..., C_q\}$ such that $\tau(\mathscr{C})$ $(= \sum_{C_i \in \mathscr{C}} \tau(C_i))$ is maximized, where $C_i$ is a community $(\subseteq V)$, $\cup_{i=1}^{q} C_i = V$, $C_i \cap C_j$ $(i \neq j)$ may or may not be empty with $1 \leq i, j \leq q$, and $q$ is the number of communities of $G$.

## 3 NP-HARDNESS

In this section, we show that the overlapping community detection problem is NP-hard, by a non-trivial reduction from the *relative density community detection problem* [27] that has been shown to be NP-hard. Since the non-overlapping community detection problem is a special case of the overlapping community detection problem, the NP-hardness of the non-overlapping community detection problem implies the NP-hardness of the overlapping community detection problem.

Let us formally define the decision versions of the relative density and non-overlapping community detection problems as follows.

***Definition 3.*** Given a graph $G = (V, E)$ and a positive rational number $\rho$ with $0 < \rho \leq 1$, the decision version of the *Relative Density Community Detection (RDCD)* problem is to determine whether there is a subset of vertices $C \subset V$ such that $e(C)/(e(C) + e(C, V \setminus C)) \geq \rho$.

***Definition 4.*** Given a graph $G = (V, E)$ and a positive rational number $\epsilon > 0$, the decision version of the *Non-overlapping Triangle Community Detection (NTCD)* problem is to determine whether there is a subset of vertices $C \subset V$ such that $|\Delta_G(C)|/(|C| + |\Delta_G(C, V \setminus C)|) \geq \epsilon$.

***Definition 5.*** Given a graph $G = (V, E)$ and a positive rational number $\epsilon' > 0$, the decision version of the *Simplified Non-overlapping Triangle Community Detection (SNTCD)* problem is to determine whether there is a subset of vertices $C \subset V$ such that $|\Delta_G(C)|/|\Delta_G(C, V \setminus C)| \geq \epsilon'$.

Note that the SNTCD problem is similar to the NTCD problem, in a sense that only $|C|$ is omitted from the denominator of the fitness metric, which makes the SNTCD problem easier than the NTCD problem. The following lemma states that the NTCD problem can be reduced to SNTCD problem in polynomial time.

***Lemma 2.*** The NTCD problem can be reduced to SNTCD problem in polynomial time.

*Proof:* One can transform a polynomial time solution to the decision version of SNTCD into a polynomial time solution for the optimization version of SNTCD by binary search on the bound $\epsilon$, and determine the set $C$ of vertices in the optimal solution in polynomial time. The algorithm for finding the set $C$ is as follows.

The algorithm proceeds iteratively. Within each iteration, it removes an edge $e \in E$ from $G$ and checks if there is a subset of vertices $C \subset V$ in the resulting graph such that $|\Delta_G(C)|/(|C| + |\Delta_G(C, V \setminus C)|) \geq \epsilon + 1/n^2$. Having removed edge $e$ from $G$, if there is still a subset of vertices $C \subset V$ such that $|\Delta_G(C)|/(|C|+|\Delta_G(C, V \setminus C)|) \geq \epsilon+1/n^2$ in the resulting graph, then $e$ is a cut edge (one of its endpoints is in $C$); otherwise, $e$ is a community edge (both of its endpoints are in $C$). If there is no subset $C$ such that $|\Delta_G(C)|/(|C| + |\Delta_G(C, V \setminus C)|) \geq \epsilon + 1/n^2$, but there is a subset $C$ such that $|\Delta_G(C)|/(|C|+|\Delta_G(C, V \setminus C)|) \geq \epsilon$, then $e$ is neither a community edge nor a cut edge, it should be removed from $G$. This procedure continues until all edges in $G$ are examined. Therefore, given a polynomial time algorithm for the NTCD problem, the SNTCD problem can also be solved in polynomial time. That is, given an instance of the SNTCD problem and $\epsilon' = p/q$, the NTCD problem can be solved, using different values of $\ell$ with $1 \leq \ell \leq n$, i.e., $|\Delta_G(C)|/(|C| + |\Delta_G(C, V \setminus C)| = p/(\ell + q)$, then determine the set $C$ and check if $|C| = \ell$. □

The following theorem shows that the SNTCD problem is NP-complete by a reduction from the RDCD problem.

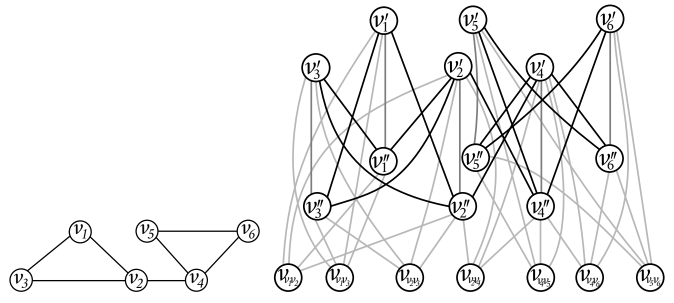
***Theorem 1.*** The simplified non-overlapping triangle community detection problem (SNTCD) is NP-complete.

*Proof:* We first show that SNTCD belongs to NP. Given a graph $G = (V, E)$, a positive rational number $\epsilon'$ and a certificate $C \subset V$, we can count the number of triangles within $C$, i.e. $|\Delta_G(C)|$, and the ones that have two vertices in $C$, i.e. $|\Delta_G(C, V \setminus C)|$. We then check if $|\Delta_G(C)|/|\Delta_G(C, V \setminus C)| \geq \epsilon'$. Thus, SNTCD is in NP.

We then show that SNTCD is NP-hard, using a polynomial time reduction from the RDCD problem. Given an instance of the RDCD problem: a graph $G = (V, E)$ and a positive rational number $\epsilon' \leq 1$, we construct an instance of the SNTCD problem containing a graph $G' = (V', E')$ and a positive rational number $\rho = 4\epsilon'/(1-\epsilon')$ in polynomial time that determines the RDCD problem in polynomial time.

Given a graph $G = (V, E)$ and $\epsilon' > 0$, we construct a graph $G' = (V', E')$, where the set $V'$ of vertices contains $2n + m$ vertices that consist of two vertices $v'$ and $v''$ for every vertex $v \in V$, and a vertex $v_{v_i,v_j}$ for every edge $(v_i, v_j) \in E$ (note that the graph is undirected, therefore $(v_i, v_j)$ and $(v_j, v_i)$ refer to the same edge and we make no distinction between them). The set of edges $E'$ contains $n + 4m$ edges that consist of an edge $(v', v'')$ for every vertex $v \in V$, and four edges $(v'_i, v_{v_i,v_j})$, $(v''_i, v_{v_i,v_j})$, $(v'_j, v_{v_i,v_j})$ and $(v''_j, v_{v_i,v_j})$ for every edge $(v_i, v_j) \in E$. From the construction of graph $G'$, it is implied that for every edge $(v_i, v_j) \in E$, there is exactly four triangles in $G'$ ($\Delta_{(v'_i,v''_j,v_{v_i,v_j})}$, $\Delta_{(v''_i,v'_j,v_{v_i,v_j})}$, $\Delta_{(v'_i,v''_i,v_{v_i,v_j})}$, and $\Delta_{(v'_j,v''_j,v_{v_i,v_j})}$). Furthermore, the number of triangles in $G'$ is exactly $4m$. Fig. 2 illustrates an example of a reduction from graph $G$ to graph $G'$.

Given a graph $G = (V, E)$ and $\epsilon' > 0$, there is a subset $C \subset V$ of vertices such that $e(C)/(c(C) + e(C, V \setminus C)) \geq \epsilon'$, if and only if, in graph $G' = (V', E')$ (constructed as described), there is a subset $C' \subset V'$ of vertices such that $|\Delta_{G'}(C')|/|\Delta_{G'}(C', V' \setminus C')| \geq 4\epsilon'/(1 - \epsilon')$. Assume that in graph $G = (V, E)$, there exists a subsets of vertices $C$ such that $e(C)/(c(C) + e(C, V \setminus C)) \geq \epsilon'$. Consider a subset of vertices $C' \subset V'$ that consists of $2|C| + e(C)$ vertices, including vertices $v', v''$ for each vertex $v \in C$, and $v_{v_i,v_j}$ for every edge $(v_i, v_j) \in E(C)$. It can be seen that $|\Delta_{G'}(C')| = 4e(C)$, we thus have $|\Delta_{G'}(C')|/|\Delta_{G'}(C', V \setminus C')| = 4e(C)/|\Delta_{G'}(C', V' \setminus C')|$.



(a) $G$ (b) $G'$

Fig. 2. $G'$ is constructed from $G$ in polynomial time. For every vertex $v_i \in V$, there are two vertices $v_i', v_i''$ in $V'$ and for every edge $(v_i, v_j) \in E$, there is one vertex $v_{v_i,v_j}$ in $V'$. Every vertex $v_{v_i,v_j} \in V'$ is connected to 4 vertices $v_i', v_i'', v_j', v_j''$, and every $v_i'$ is connected to $v_i''$.

oreover, for every edge $(v_i, v_j) \in E(C, V \setminus C)$, there is only one triangle $\Delta_{(v_i', v_i'', v_{v_i,v_j})}$ with two endpoints in $C'$, we have

$$\begin{aligned}
&|\Delta_{G'}(C')|/|\Delta_{G'}(C', V \setminus C')| \\
&= 4(e(C)/e(C, V \setminus C)) \\
&= 1/(e(C, V \setminus C)/e(C) + e(C)/e(C) - 1)
\end{aligned}$$

Since we assumed that $e(C)/(e(C) + e(C, V \setminus C)) \geq \epsilon'$,

$$|\Delta_{G'}(C')|/|\Delta_{G'}(C', V \setminus C')| \quad \geq \quad 4/(1/\epsilon' - 1) = \rho.$$

Now, assume that in graph $G'$ there is a subset $C' \subset V'$ of vertices such that $|\Delta_{G'}(C')|/|\Delta_{G'}(C', V \setminus C')| \geq 4\epsilon'/(1-\epsilon')$, we show that there is a subset $C \subset V$ of vertices such that $e(C)/(e(C) + e(C, V \setminus C)) \geq \epsilon'$. First, it is noted that if a vertex $v'$ is in $C'$, then its copy $v''$ is also in $C'$. For every triangle formed by $v'$ and vertices within $C'$, there is at least one triangle formed by vertices in $C'$ and $v''$. Therefore, if $v''$ is excluded from $C'$, the number of cut triangles will be larger than the number of triangles formed by vertex $v'$ and as a result, removing vertex $v'$ from $C'$ will increase the fitness value of $C'$. Therefore, if $v'$ is in $C'$, then $v''$ is also in $C'$. Using a similar reasoning, it is also implied that if $v_{v_i,v_j}$ is in $C'$, then vertices $v_i', v_i'', v_j'$, and $v_j''$ lie in $C'$.

We finally show that if there is a subset $C' \subset V'$ such that $|\Delta_{G'}(C')|/|\Delta_{G'}(C', V' \setminus C')| \geq 4\epsilon'/(1-\epsilon')$, then there is a subset $C \subset V$ of vertices such that $e(C)/(e(C)+e(C, V \setminus C)) \geq \epsilon'$. For every vertex $v_{v_i,v_j} \in C'$, add both vertices $v_i$ and $v_j$ to $C$. As a result, the number of edges in $E(C)$ is $|\Delta_{G'}(C')|/4$, and the number of cut edges is equal to the number of cut triangles in $C'$. Therefore,

$$\begin{aligned}
&e(C)/(e(C) + e(C, V \setminus C)) \\
&\quad = \quad 4|\Delta_{G'}(C')|/(4|\Delta_{G'}(C')| + |\Delta_{G'}(C', V' \setminus C')|) \\
&\quad = \quad (1 + |\Delta_{G'}(C', V' \setminus C')|/4|\Delta_{G'}(C')|)^{-1},
\end{aligned}$$

since $|\Delta_{G'}(C')|/|\Delta_{G'}(C', V' \setminus C')| \geq 4\epsilon'/(1-\epsilon')$,

$$e(C)/(e(C) + e(C, V \setminus C)) \quad \geq \quad (1 + (1-\epsilon')/\epsilon')^{-1} \geq \epsilon'.$$

Hence, the RDCD problem can be reduced to the SNTCD problem in polynomial time. As the RDCD problem is NP-complete, the SNTCD problem is NP-complete, too. □

## 4 Algorithm

In this section, we first propose an efficient yet scalable algorithm for the overlapping community detection problem, which will deliver a feasible solution. We then show the properties of the overlapping communities delivered by the proposed algorithm and analyze the time complexity of the proposed algorithm.

### 4.1 Algorithm description

To identify high-quality overlapping communities in $G$, the algorithm consists of two stages. It detects non-overlapping core communities, using the proposed community fitness metric $\tau(\cdot)$. Notice that the core communities are exclusive to each other, they are the bases to form overlapping communities. It then expands the core communities to form overlapping communities.

In the core community detection, the vertices in $G(V, E)$ is partitioned into core communities so that each core community is a densely connected subgraph, using the fitness metric $\tau(\cdot)$. It is noticed that these core communities are exclusive to each other. Let $\mathscr{C}$ be the set of core communities whose construction proceeds iteratively. Initially, there is only one single community including all the vertices in $G$, i.e., $\mathscr{C} = \{V\}$. Within iteration $k$ ($k \geq 1$), some of the edges in $G$ will be removed if the support of an edge is no more than $k$. The edge removal will increase the value of the fitness metric of the resulting connected components. Specifically, for each community $C \in \mathscr{C}$ in iteration $k$, let $\Phi_k^C$ be the edges in the induced subgraph $G[C]$ with support no greater than $k$, the set $\Phi_k^C$ will be examined to check if its removal can increase the value of the fitness metric of the resulting communities. If yes, the edges in $\Phi_k^C$ are removed from $G$ and community $C$ is replaced by the number of connected components derived from it. Notice that the support of each remaining edge in the resulting graph will be updated accordingly if the edges in $\Phi_k^C$ are removed from $G$. The value of $k$ is then incremented by one after each iteration. This procedure continues until the support of each edge in the resulting graph is no less than $k$.

Having found all core communities of $G$, the core community expansion then follows, by adding vertices from other communities to each core community greedily. Specifically, given the set of core communities $\mathscr{C} = \{C_1, ..., C_q\}$, let $\tau(C_i)$ be the overlapping triangle connectivity fitness value of community $C_i$ for all $i$ with $1 \leq i \leq q$. The core community expansion finds a collection $\mathscr{C}$ of overlapping communities, which are local maxima communities according to the fitness metric $\tau(\cdot)$. The core community expansion of each community $C \in \mathscr{C}$ proceeds iteratively, by adding a neighbor $v \notin C$ of a vertex in $C$ such that the value of $\tau(C \cup \{v\})$ is the maximum one among all the other neighbors. This iteration is repeated until no such a neighbor can be added to the expanded $C$. The detailed procedure for this is as follows. Let $N_C$ be the set of neighbors of community $C$ that their additions to $C$ can increase the fitness value of community $C$. The algorithm finds such a vertex $v \in N_C$ that its addition to $C$ increases the fitness value of $C \cup \{v\}$ more than the other vertex $v' \in N_C \setminus \{v\}$ in $N_C$, and $v$ is added to community $C$. Note that only a vertex $v$ is added to community $C$ each time and the set $N_C$ of neighbors then will be updated accordingly. This iterative procedure is repeated for every community $C \in \mathscr{C}$ until there is not any neighbor in $N_C$ that can increase the fitness value of $C$.

It must be mentioned that although the first stage of the proposed algorithm exhibits some similarities with traditional $k$-truss detection algorithm [4], they are essentially different. Specifically, the $k$-truss detection algorithm repeatedly removes edges with support no larger than $k$ for a given $k$, while the proposed algorithm here starts with $k = 1$ and increments $k$ in each iteration until the sum of the fitness values of all detected communities cannot be further increased. In contrast, traditional $k$-truss algorithms repeatedly remove edges with support no larger than $k$, regardless of the sum of the fitness values of the resulting communities [4].

The detailed algorithm for the overlapping community detection problem is given in `Algorithm` 1.

**Algorithm 1** Overlapping_Community_Detection($G$)

```
Input: $G = (V, E)$
Output: Overlapping communities $\mathscr{C}$ of $G$
 1: /* $\mathscr{C}$ is the collection of detected communities */
 2: $\mathscr{C} \leftarrow \{V\}$;
 3: $k \leftarrow 0$;
 4: /* Find core communities */
 5: Calculate the support of each edge in $E$, using triangle
    counting algorithm in [16];
 6: while $\exists e \in E$ with $sup_G(e) \geq k$ do
 7:     for each community $C \in \mathscr{C}$ do
 8:         $\Phi_k^C \leftarrow \{e \mid edge\ e \in G[C]$ with $sup_G(e) \leq k\}$;
 9:         $\mathscr{C}' \leftarrow \{C' \mid C'$ is a connected component in
    $G[C] \setminus \Phi_k^C\}$;
10:         if $\tau(C) \leq \tau(\mathscr{C}')$ then
11:             Remove the edges in $\Phi_k^C$ from network $G$;
12:             Replace $C$ in $\mathscr{C}$ with communities in $\mathscr{C}'$;
13:     /* Increment the value of $k$ by 1 */
14:     $k \leftarrow k + 1$;
15: /* Expand core communities in $\mathscr{C}$ to form overlapping
    communities */
16: for each core $C \in \mathscr{C}$ do
17:     $N_C \leftarrow$ Neighbors of $C$, whose addition does not
    decrease the fitness value of $C$;
18:     while $N_C \neq \emptyset$ do
19:         $v \leftarrow \text{argmax}_{u \in N_C}\{\tau(C \cup \{u\})\}$;
20:         $N_C \leftarrow N_C \setminus \{v\}$ /* remove vertex $v$ from $N_C$ */;
21:         $C \leftarrow C \cup \{v\}$ /* add vertex $v$ to $C$ */;
22:         for each neighbor $u$ of vertex $v$ do
23:             if $\tau(C \cup \{u\}) \geq \tau(C)$ then
24:                 Add the neighbor $u$ to $N_C$;
    return $\mathscr{C}$;
```

## 4.2 An example of the algorithm execution

We use an example to illustrate the execution of the proposed algorithm, `Algorithm` 1. Fig. 3 shows the execution results of `Algorithm` 1 on an input social network at different stages.

Fig. 3(a)-Fig. 3(c) illustrate the results of core community detections in stage one, where the edges with low support (red dashed edges) in Fig. 3(a) and Fig. 3(b) are removed until no edge is left, while Fig. 3(d)-Fig. 3(h) show the results of core community expansion in stage two, where community $B$ in Fig. 3(d) is expanded by adding one extra neighbor (the green vertex) into it, the rest of neighbors of the community will not be added, since they would not increase the fitness value of the expanded community. Similarly, in Fig. 3(f)-3(h) community $A$ is expanded by adding more green neighbors, since adding them to the community will increase its fitness value.

It can be seen in Fig. 3(a) that all edges with support no larger than $k = 1$ are removed, as their removal results in the increase in the fitness value of the resulting communities. Similarly, in Fig. 3(b), when $k = 2$, those edges with support no larger than two are removed. However, Fig. 3(c) shows that the removal of the edges with support no larger than $k = 3$ does not increase the fitness value of communities, therefore, these edges will not be removed in this iteration. It is also noticed that core communities obtained in the first stage are not $k$-trusses for $k = 3$. This demonstrates the difference between traditional $k$-truss detection algorithms [4] and this algorithm for core community detections. Fig. 3(d)-3(h) depict the expansion of communities. Fig. 3(d)-Fig. 3(e) illustrate that the addition of one vertex to community $B$ increases the fitness value of $B$. Similarly, Fig. 3(f)-3(h) show that the addition of two vertices to community $A$ increases the fitness value of $A$.

## 4.3 Algorithm analysis

The rest is to show the properties of overlapping communities delivered by `Algorithm` 1 and analyze the time complexity of the proposed algorithm as follows.

We first prove that the overlapping communities delivered by `Algorithm` 1 do not have the separation effect if certain conditions are met.

***Lemma 3.*** Given a social network $G = (V, E)$ and two core communities $C_1 \subseteq V$ and $C_2 \subseteq V$ obtained in the first stage of `Algorithm` 1, if there is a set of vertices $C_2' \subseteq C_2$ in which every vertex forms at least one triangle with the vertices in $C_1$, $C_2'$ will be assigned to both $C_1$ and $C_2$ in the overlapping communities found after the second stage of `Algorithm` 1.

*Proof:* Consider two core communities $C_1$ and $C_2$ of $G$. Let $C_2'$ be a subset of $C_2$ in which each vertex $v \in C_2'$ forms at least one triangle with the other two vertices in $C_1$, as illustrated by Fig. 4. The vertices in $C_2'$ then will be assigned to $C_1$ by the fitness metric $\tau(\cdot)$, i.e., $\tau(C_1 \cup C_2') \geq \tau(C_1)$, where $C_2'$ is a subset of community $C_2$ such that each vertex in $C_2'$ forms at least one triangle with the vertices in $C_1$.

Let $t$ be the number of triangles formed by the vertices in $C_2'$ and $C_1$, and let $t'$ be the number of triangles formed by the vertices in $C_1$ and the vertices in $V \setminus C_1$. We show that the vertices in $C_2'$ will be assigned to $C_1$ in the second stage

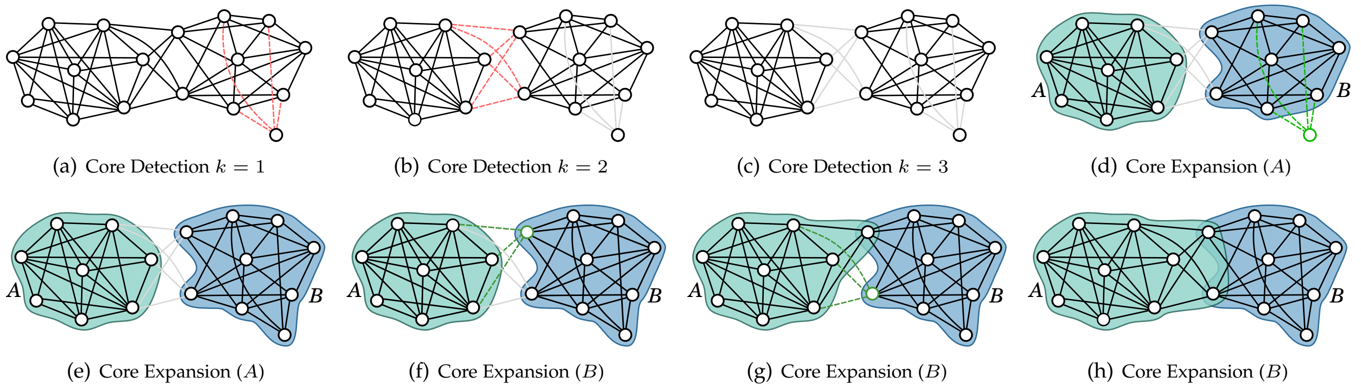

(a) Core Detection $k = 1$ (b) Core Detection $k = 2$ (c) Core Detection $k = 3$ (d) Core Expansion $(A)$ (e) Core Expansion $(A)$ (f) Core Expansion $(B)$ (g) Core Expansion $(B)$ (h) Core Expansion $(B)$

Fig. 3. A running example of the proposed algorithm. In the core community detection phase, the network is partitioned into core communities and these communities are expanded in the core community expansion phase, where the overlapping region between two communities is detected.

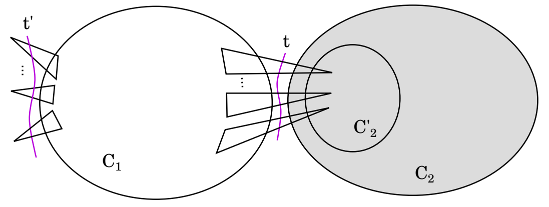


Fig. 4. Given two communities $C_1$ and $C_2$, a subset $C'_2 \subset C_2$ forms many triangles with the vertices in $C_1$.

of Algorithm 1 by contradiction. Assume that vertices in $C'_2$ will not be assigned to $C_1$, i.e., $\tau(C_1 \cup C'_2) < \tau(C_1)$, then,

$$\tau(C_1 \cup C'_2) < \tau(C_1) \Rightarrow \frac{|\Delta_G(C_1 \cup C'_2)|}{|C_1 \cup C'_2| + t'} < \frac{|\Delta_G(C_1)|}{|C_1| + t + t'}. \tag{4}$$

Since every vertex in $C'_2$ forms at least one triangle with the vertices in $C_1$, we have $t > |C'_2|$,

$$\frac{|\Delta_G(C_1 \cup C'_2)|}{|C_1 \cup C'_2| + t'} < \frac{|\Delta_G(C_1)|}{|C_1| + |C'_2| + t'}. \tag{5}$$

Since the vertices in two communities $C_1$ and $C_2$ are vertex-disjoint, we have

$$C_1 \cap C'_2 = \emptyset \Rightarrow |C_1 \cup C'_2| = |C_1| + |C'_2|.$$

Thus,

$$\frac{|\Delta_G(C_1 \cup C'_2)|}{|C_1| + |C'_2| + t'} < \frac{|\Delta_G(C_1)|}{|C_1| + |C'_2| + t'} \Rightarrow |\Delta_G(C_1 \cup C'_2)| < |\Delta_G(C_1)|. \tag{6}$$

Inequality (6) does not hold, otherwise this leads to a contradiction that $|\Delta_G(C_1 \cup C'_2)| \geq |\Delta_G(C_1)| + t$. Therefore, the vertices in $C'_2$ will be assigned to $C_1$ in the second stage of Algorithm 1, since $\tau(C_1 \cup C'_2) \geq \tau(C_1)$. □

We then show that the community fitness metric $\tau(\cdot)$ avoids free rider effect if a certain condition meets by the following lemma.

***Lemma 4.*** Given a social network $G = (V, E)$ and two core communities $C_1 \subseteq V$ and $C_2 \subseteq V$ found in the end of the first stage of Algorithm 1, if there is a set of vertices $C'_2 \subseteq C_2$ in which every vertex forms at least one triangle with the vertices in $C_1$, community $C_1$ and $C_2$ will not be merged in the second stage of Algorithm 1 unless $\tau(C_1) \leq \tau(C_1 \cup C_2)$ and $\tau(C_2) \leq \tau(C_1 \cup C_2)$.

*Proof:* We show that the community fitness metric $\tau(\cdot)$ avoids free rider effect if a certain condition is met by contradiction. Considering the proof for Lemma 3, we need to show that $\tau(C'_2) \leq \tau(C_1 \cup C'_2)$ to avoid the free rider effect on the overlapping communities found in the second stage of Algorithm 1. Without loss of generality, we assume that $\Delta_G(C_1)/|C_1| > \Delta_G(C'_2)/|C'_2)|$. Assume that $\tau(C'_2) > \tau(C_1 \cup C'_2)$,

$$\tau(C_1 \cup C'_2) < \tau(C'_2) \Rightarrow \frac{|\Delta_G(C_1 \cup C'_2)|}{|C_1| + |C'_2| + t'} < \frac{|\Delta_G(C'_2)|}{|C'_2| + t' + t''}$$

where $t''$ is the number of triangles that have two vertices in $C'_2$ and one vertex outside $C'_2$,

$$\frac{|\Delta_G(C_1)| + |\Delta_G(C'_2)| + t}{|C_1| + |C'_2| + t'} < \frac{|\Delta_G(C'_2)|}{|C'_2| + t' + t''}$$
$$(|\Delta_G(C_1)| + |\Delta_G(C'_2)| + t)(|C'_2| + t' + t'') < (|\Delta_G(C'_2)|)(|C_1| + |C'_2| + t')$$

Since $t'' > t$,

$$(|\Delta_G(C_1)| + t)(|C'_2| + t' + t'') + \Delta_G(C'_2)t'' < (|\Delta_G(C'_2)|)|C_1|,$$
$$(|\Delta_G(C_1)| + t)(|C'_2| + t' + t'') < (|\Delta_G(C'_2)|)(|C_1| - t''),$$
$$\frac{|\Delta_G(C_1)| + t}{|C_1| - t''} < \frac{|\Delta_G(C'_2)|}{|C'_2| + t' + t''},$$

which is a contradiction to the initial hypothesis that $\Delta_G(C_1)/|C_1| > \Delta_G(C'_2)/|C'_2)|$. The lemma thus holds. □

In summary, we have the following theorem.

***Theorem 2.*** Given a social network $G = (V, E)$, Algorithm 1 for the overlapping community detection problem will deliver a feasible solution in time $O(|V| \cdot |E|)$, if the support of each edge is constant.

*Proof:* Following Algorithm 1, the core community detection in $G$ starts with calculating the support of each edge of $G$ in time $O(|E|^{3/2})$, using the algorithm in [16], and it proceeds iteratively. Within each iteration, the set $\Phi_k = \bigcup_{C_i \in \mathscr{C}} \Phi_k^C$ of edges with support no larger than $k$ is found, using the values calculated at its step 4. It then calculates the fitness scores of the resulting connected components in $G[C] \setminus \Phi_k^C$ in linear time. Each iteration of the while-loop of Algorithm 1 takes $O(|E|)$ time. Since the maximum support among edges in a real social network usually is constant, the number of iterations $k$ is constant. The time spent on core community detection by Algorithm 1 is $O(|E|^{3/2})$.

In the core community expansion stage of Algorithm 1, each vertex $v$ is added to set $C$ at most once. Initially, neighbors of community $C$ are added to set $N_C$, then a vertex $u \in N_C$ with the maximum value $\tau(C \cup \{u\})$ is added to $C$. When a vertex $v$ is added to $N_C$, the value of $\tau(N_C \cup \{v\})$ is calculated, by finding the number of triangles formed by every edge in $E[C]$ connected to $v$ in $O(|E[C]|)$ time, assuming that a heap data structure is adopted for keeping track of vertices in $N_C$. Then, the insertion and extraction of vertices in $N_C$ (represented by the heap data structure) takes $O(|V| \cdot \log |V|)$ time. Thus, overlapping community identification derived from the found core communities takes $O(q \cdot |E|)$ time if there are $q$ core communities. While $q \leq |V|$, the time complexity of Algorithm 1 thus is $O(|E|^{3/2} + q \cdot |E|) = O(|E|^{3/2} + |V| \cdot |E|) = O(|V| \cdot |E|)$. □

It must be mentioned that the analytical time complexity of Algorithm 1 is very conservative. Its actual running time on real social networks is much faster, which is almost linear to the problem size $|V| + E|$, due to the sparsity of social networks. This can be witnessed from later empirical evaluation results (see Fig. 6).

## 5 Experimental Results

In this section, we evaluate the performance of the proposed algorithm against several benchmark algorithms using several real-world datasets. We also study separation and free rider effect on the overlapping communities delivered by different algorithms under different fitness metrics including the one proposed in this paper.

### 5.1 Experimental environment settings

We start with the experimental environment settings and descriptions of different data sets, evaluation metrics, and benchmark algorithms.

**Benchmark algorithms.** To evaluate the performance of the proposed algorithm, Algorithm 1, denoted by CoreExp, for the overlapping community detection problem, the following state-of-the-arts will be adopted for the benchmark purpose.

- FOCS [2] – A local expansion algorithm, which finds communities starting from neighborhoods of vertices. This algorithm expands the initial communities by adding vertices using the local modularity fitness metric.
- Demon [5] – An agent-based algorithm, in which, every vertex $v$ receives a label $l$, where $l$ is the label appeared in majority of neighbors of $v$. The labels are propagated iteratively until every vertex has the label of most of its neighbors. Finally, communities that have more than a certain overlap are merged.
- Bigclam [35] – A matrix factorization-based algorithm, which assigns each vertex-community a value that represents the membership in the community. It then models the probability of an edge as a function of the shared community affiliations and identifies network communities by fitting the model to the given network, and estimating the latent factors.
- SeedExp [30] – A local expansion algorithm, which consists of four phases: filtering, seeding, seed expansions, and propagations. The filtering phase removes weakly connected subgraphs. The seeding phase finds seed vertices, which are expanded in the seed expansion phase. The propagation attaches the weak components to communities.
- Bayes [11] – A Bayesian model-based algorithm, which posits a probabilistic model of networks where each vertex can belong to many communities. It finds the conditional distribution of the hidden communities given the observed network. It then approximates the conditional distribution with various methods in combination with stochastic optimization by iteratively subsampling the network and estimation of the hidden communities.

TABLE 1
Details of real datasets, where $\mathscr{C}^*$ represents the set of ground-truth communities and $sup_{\max}$ is the largest support of edges.

| **Dataset** | $\|V\|$ | $\|E\|$ | $\|\mathscr{C}^*\|$ | $sup_{\max}$ |
|---|---|---|---|---|
| Facebook | 4,039 | 88,234 | 193 | 293 |
| Twitter | 81,306 | 2,420,766 | 4,065 | 9,016 |
| Google Plus | 107,614 | 30,494,866 | 468 | 11,488 |
| Amazon | 334,863 | 925,872 | 253,345 | 161 |
| DBLP | 317,080 | 1,049,866 | 13,477 | 213 |
| Orkut | 3,072,441 | 117,185,083 | 15,301,901 | 9,145 |
| LiveJournal | 3,997,962 | 34,681,189 | 658,401 | 1,393 |

**Real datasets.** We make use of seven real datasets, which have been widely adopted in literature [32] and are publicly available[1]. Specifically, dataset Amazon is based on the Amazon products, where there is an edge between products $i$ and $j$ if product $i$ is frequently co-purchased with product $j$, and each product category provided by Amazon is a ground-truth community. Dataset DBLP is a collaboration network, where ground-truth communities are defined as publication venues, e.g., journals or conferences. Dataset Orkut is the friendship network of Orkut members, in which communities are the groups that users create and other users join in. Dataset LiveJournal is the friendship network of users in LiveJournal blogging web site. Users can define groups and join multiple groups. These groups are considered as the ground-truth communities. Dataset Facebook consists of ego networks of Facebook users, which has been collected from survey participants. The groups provided by users are the ground-truth communities. Dataset Twitter consists of 'lists' from Twitter. The social communities are the ground-truth communities in Twitter. Dataset Google Plus is a social network in Google+. The groups that are defined by users represent ground-truth communities. Notice that each of the datasets Facebook, Twitter and Google Plus in fact is a combined network consisting of ego networks from SNAP. Table 1 details the mentioned datasets in our experiments.

**Evaluation measures.** quantitatively measuring the quality of detected overlapping communities in a social

1. http://snap.stanford.edu/data/index.html

network is challenging, as different measures lead to different quality of overlapping communities. We here employ two widely-adopted measures [10], [11], [30], [32], [35] for analyzing the accuracy of the detected communities by different algorithms, i.e., $F_1$ and $F_2$-measures [32].

Let $\mathscr{C}^*$ be the set of ground-truth communities and $\mathscr{C}$ the detected communities by any mentioned algorithm. The F-measure is based on the precision and recall of each community $C$ compared to $C^*$ defined as follows.

$$p(C, C^*) = \frac{|C \cap C^*|}{|C|}, \quad r(C, C^*) = \frac{|C \cap C^*|}{|C^*|}$$

$F_1$-measure [32] is the harmonic mean of the precision and recall, while $F_2$-measure [32] magnifies the impact of recall in the results, as follows.

$$F_1 = \frac{1}{|\mathscr{C}|} \sum_{C \in \mathscr{C}} \max_{C^* \in \mathscr{C}^*} \{\frac{2 \cdot p(C, C^*) \cdot r(C, C^*)}{p(C, C^*) + r(C, C^*)}\}, \tag{7}$$

$$F_2 = \frac{1}{|\mathscr{C}|} \sum_{C \in \mathscr{C}} \max_{C^* \in \mathscr{C}^*} \{\frac{5 \cdot p(C, C^*) \cdot r(C, C^*)}{4 \cdot p(C, C^*) + r(C, C^*)}\}. \tag{8}$$

Note that all the experiments are conducted on a desktop with a 3.06GHz CPU and 16GB memory.

## 5.2 Performance evaluation of different algorithms

We first study the performance of the proposed algorithm `CoreExp` against the benchmark algorithms, by evaluating the quality of the found communities under two measures: $F_1$-measure and $F_2$-measure. We also compare the running times of different algorithms and the number of communities delivered by each of them. Fig. 5 plots the quality bars of the communities delivered by different algorithms, using different datasets and community fitness metrics. Notice that algorithm `Demon` did not terminate after 48 hours for datasets `LiveJournal` and `Orkut`. The results on those datasets are not available, and thus cannot be shown in the bar chart.

Fig. 5 shows that algorithm `CoreExp` delivers the most accurate communities for most datasets (6 out of the 7 datasets). Specifically, it can be seen from Fig. 5(a) that for dataset `Facebook`, algorithm `CoreExp` outperforms all other algorithms at least by 10% in both $F_1$ and $F_2$-measures. Similarly, Fig. 5(b) demonstrates that algorithm `CoreExp` outperforms all other algorithms by at least 12% in the ego network of dataset `Twitter`. Fig. 5(c) indicates that algorithm `CoreExp` outperforms all other algorithms by at least 15% for dataset `Google Plus`. For the dataset `Amazon`, Fig. 5(d) shows that algorithm `CoreExp` is superior to other mentioned algorithms by at least 10% under both $F_1$-measure and $F_2$-measure. It can also be seen from Fig. 5(e) that for the dataset `DBLP`, algorithm `CoreExp` outperforms the other algorithms by at least 11% based on both $F_1$ and $F_2$ measures. It is noticed that for dataset `LiveJournal` Fig. 5(f) indicates that both algorithm `Bigclam` and algorithm `FOCS` beat algorithm `CoreExp`. This may partially contribute to the specific topological structure of the LiveJournal network. However, the better quality solution delivered by algorithm `Bigclam` is at the expense of prohibitive running time that is several orders of magnitudes of that of algorithm `CoreExp`. On the other hand, despite that algorithm `FOCS` has a less running time compared with algorithm `CoreExp`, the communities delivered by algorithms `Bigclam` and `FOCS` will cause separation and free rider effects that can be seen from Table 2, while the communities delivered by algorithm `CoreExp` can avoid such effects. Fig. 5(g) demonstrates that algorithm `CoreExp` is the best one among all comparison algorithms which has the highest accuracy of communities found for dataset `Orkut`. The reason behind is that dataset `Orkut` contains more than 100 million edges and identifying communities from such a large-scale network is challenging. Algorithm `CoreExp` demonstrates that even for such a massive dataset, it outperforms the other algorithms significantly in terms of both the quality of the solution and the running time (see Fig. 6).

Fig. 6 plots the running times of different algorithms. It is noticed that algorithm `CoreExp` takes a few hours for the massive dataset such as `Orkut` with more than 100 million edges and less than one hour for dataset `LiveJournal` with more than 34 million edges. It can be observed from Fig. 6 that the running time of algorithm `CoreExp` is at least 10% less than that of all other algorithms except that of algorithms `Bayes` and `FOCS`. However, the quality of the solutions delivered by algorithms `Bayes` and `FOCS` is not as good as that by algorithm `CoreExp`. colorblue Similarly, for dataset `Orkut`, the running time of algorithm `SeedExp` is the smallest, while the communities obtained by `SeedExp` have the poor quality for this dataset.

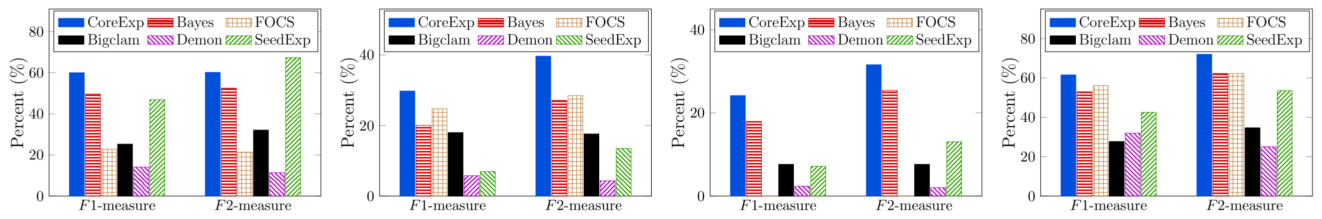
Fig. 7 depicts the number of communities delivered by each mentioned algorithm, from which, it can be seen that algorithm `CoreExp` delivers more communities than that of any other mentioned algorithm except algorithms `Bayes` and `FOCS`. However, the quality of the solution delivered by algorithm `CoreExp` is better than either by algorithms `Bayes` and `FOCS` in most cases.

## 5.3 Separation and free rider effects on the communities found by different algorithms

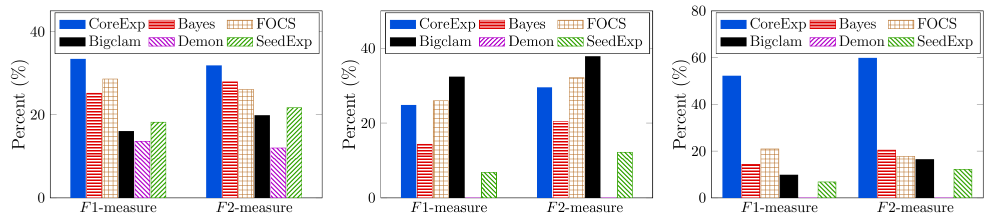
We then evaluate separation and free rider effects on the communities found by different algorithms, using a synthetic dataset, as there is not any available information of these two effects on real datasets. To examine whether the communities delivered by different algorithms cause separation and free rider effects, we generate a synthetic dataset based on SSCA[2] synthetic networks. We first generate SSCA networks with network size from $2^{10}$ to $2^{20}$, where each SSCA network consists of a set of cliques. We then place a vertex $v_{ij}$, if there is an edge between two cliques $C_i$ and $C_j$, and connect vertex $v_{ij}$ to each vertex in cliques $C_i$ and $C_j$ with a probability 0.5. Vertex $v_{ij}$ is then treated as an overlapping vertex between communities $C_i$ and $C_j$, i.e, it will belong to both overlapping communities derived from $C_i$ and $C_j$, respectively. Note that the synthetic networks randomly generated may not be connected, and algorithm `SeedExp` thus is inapplicable in this case.

Table 2 lists the experimental results of free rider and separation effects of the communities delivered by different algorithms, using the constructed synthetic datasets. For each benchmark algorithm, the percent of vertices in the

2. http://www.cse.psu.edu/∼kxm85/software/GTgraph/



(a) `Facebook` (b) `Twitter` (c) `Google Plus` (d) `Amazon`



(e) `DBLP` (f) `LiveJournal` (g) `Orkut`

Fig. 5. The quality of overlapping communities found by different algorithms compared with the quality of the ground-truth communities under different community fitness metrics. Note that algorithm algorithm `Demon` did not terminate for datasets `LiveJournal` and `Orkut` after 48 hours, thus its results thus are not available in the plots.

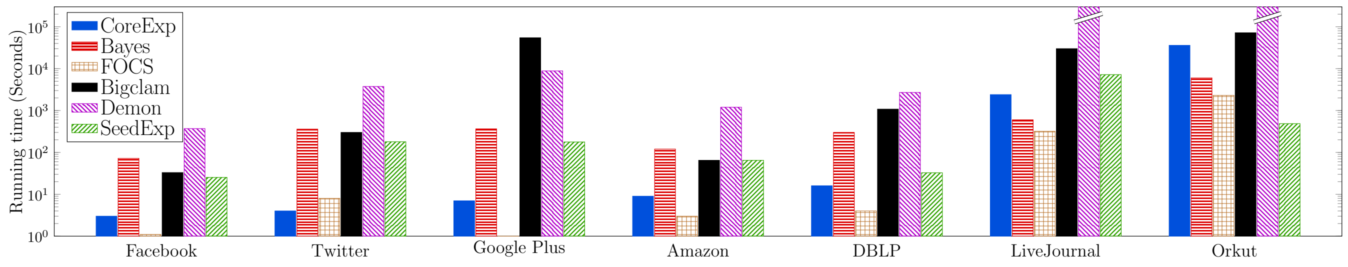


Fig. 6. The running times of different algorithms. The bars with parallel lines represent algorithms that did not terminate.

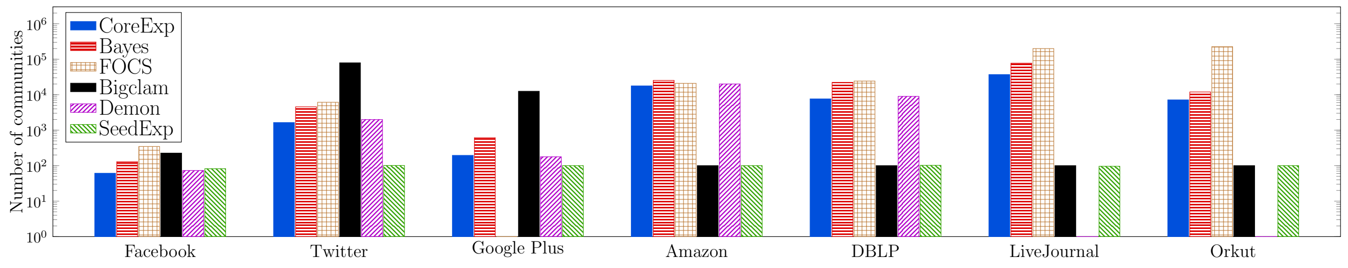


Fig. 7. The number of detected communities by different algorithms. Algorithm `Demon` did not terminate for datasets `LiveJournal` and `Orkut`, thus their results on those datasets thus will not be available from the bar chart.

overlap that were assigned to only one community represents the separation effect, and the percent of detected communities spanning more than one community in the ground-truth communities represents the free rider effect. The values reported in Table 2 are averaged across all synthetic networks.

Table 2 indicates that algorithm `CoreExp` can avoid both free rider and separation effects in these synthetic datasets. It can be observed from Table 2 that the percent for separation effect by algorithm `FOCS` is 65.320%, which means that a large number of vertices in the overlap were assigned to only one community. The reason is that algorithm `FOCS` uses the modularity fitness metric, which is prone to separation effect. Similarly, the percent of separation effect in algorithm `Bigclam` is 64.24%, which means that algorithm `Bigclam` also causes separation effect. Table 2 also shows that the percent of free rider effect caused by algorithm `Bigclam` is 18.55%, which is considerably larger than the other community detection algorithms. This number is followed by the percent of free rider effect by algorithms `Demon` and `Bayes`.

TABLE 2
Separation and free rider effects caused by different algorithms, where "–" means that algorithm does not terminate after 48 hours.

| Algorithm | Separation effect | Free rider effect |
|---|---|---|
| CoreExp | 0% | 0% |
| Bayes | 7.60% | 2.036% |
| FOCS | 65.320% | 0% |
| Bigclam | 64.24% | 18.55% |
| Demon | 4.150% | 2.330% |
| SeedExp | – | – |

While algorithm CoreExp can avoid both free rider and separation effects on these datasets, none of the benchmark algorithms can avoid both the effects.

## 6 Related Work

In recent years, considerable efforts have been taken in building efficient metrics and models that can accurately capture the properties of overlapping communities in real social networks. In their comprehensive survey, Xie *et al.* [32] surveyed state-of-the-art algorithms for overlapping community detection and categorized the algorithms into five categories: link partitioning, fuzzy community detection, agent-based algorithms, local expansion and triangle-based approaches. In the following, we survey existing algorithms for overlapping community detection.

Fuzzy community detection algorithms employ a soft clustering technique for identifying communities [32]. For example algorithm Bigclam [35] exploits a non-negative matrix factorization framework for finding overlapping communities. Nepusz *et al.* [20] modeled the overlapping community detection as a nonlinear constrained optimization problem and solved the problem using simulated annealing. One noticeable drawback of such algorithms is the need to determine the dimensionality of the community membership vector [7].

In agent-based methods, each vertex is considered as an agent that transmits messages to other vertices. For instance, each vertex in algorithm SLPA [33] can be a listener or a speaker. It spreads vertex labels across the network, and the probability of observing a vertex label in another vertex's memory determines the community membership. Algorithm Demon [5] is another agent-based method that extracts the ego network of each vertex and applies a label propagation algorithm on this structure, ignoring the presence of the ego itself, then each vertex acquires a label that appears most frequently among its neighbors to extract the communities [5]. However, agent-based algorithms are very time-consuming in real-world networks.

Local expansion methods are based on growing a community, using a community fitness metric to measure the quality of the community. Whang *et al.* [30] used a personalized PageRank algorithm for finding cuts between communities, where a random walk in a network can start from seeds only. Since the vertices close by seed are more likely to be visited, thereby receive higher ranks and join the same communities. Among the methods, Algorithm LFM [15] chooses random seeds and then expands the seeds until the value of fitness function based on the number of edges in the community is locally maximal. While the fitness metric used in the local expansion methods can capture the community density, it suffers from free rider [31] or separation effect. The proposed method in this paper falls into this category but aims to minimize free-rider and separation effects on the found communities. Bandyopadhyay *et al.* devised an algorithm called FOCS [2], where initially communities are the neighborhoods of all vertices in the network and these communities are then refined by adding and removing vertices from communities, using local modularity. However, it has been shown that both subgraph and local modularities suffer from free rider and separation effects [31].

Triangle-based approaches are particularly important in community detection [3], [4]. For instance, in triangle $k$-core motif [36], DN-Graph [29] and $k$-truss [4], the number of triangles formed by an edge is used as an indicator of its strength, where edges with fewer triangles can be removed to isolate cohesive communities. Although these approaches provide accurate non-overlapping communities, they are not capable of finding overlapping communities. Sariyuce *et al.* [26] attempted to extend $k$-trusses by proposing the $k$-$(r, s)$ nucleus ($r < s$), which is a union of $s$-cliques, where each $r$-clique is included in at least $k$ $s$-cliques and every pair of $r$-cliques are connected via a sequence of $r$-cliques. However, the experimental results in the mentioned work are limited to values $s \leq 4$, since the state-of-the-art algorithms for finding $s$-cliques are inefficient. Recently, Benson *et al.* [3] and Tsourakakis *et al.* [28] both extended the existing fitness metric *conductance* using triangle motifs. In their definition, conductance of a community is defined based on a given motif $M \subseteq G$, where the number of motif instances in a subgraph $G'$ of a social network $G$ is considered as the density, and the cut between $G'$ and other subgraphs is defined as the number of instances of motif $M$ that include some vertices from $G'$ and some from $G \setminus G'$. One of the major differences between our work in this paper and the works of Benson *et al.* and Tsourakakis *et al.* is that they consider symmetric motif cuts while we consider asymmetric motif cuts.

## 7 Conclusion

In this paper, we investigated the problem of finding high quality overlapping communities from a large social network by taking both separation and free rider effects on found overlapping communities into consideration. To this end, we first proposed a novel community fitness metric - the overlapping triangle connectivity fitness metric for overlapping community detection. We then devised an efficient yet scalable algorithm for the problem. We finally validated the effectiveness of the proposed fitness metric and evaluated the performance of the proposed algorithm, by conducting extensive experiments on seven real-world datasets. Experimental results show that the proposed algorithm is very promising and outperforms state-of-the-arts.

## Acknowledgement

We really appreciate the four anonymous referees and the Editor-in-Chief for their critical comments and constructive suggestions which helped us improve the quality and presentation of the paper. It is acknowledged that the work is fully funded by the grant of Australian Research Council Discovery Project No. DP120102627.

## References

[1] C. C. Aggarwal, J. L. Wolf, and P. S.-l. Yu. Method for targeted advertising on the web based on accumulated self-learning data, clustering users and semantic node graph techniques, Mar. 30 2004. US Patent 6,714,975.
[2] S. Bandyopadhyay, G. Chowdhary, and D. Sengupta. Focs: Fast overlapped community search. *TKDE'15*, 27(11):2974–2985, 2015.
[3] A. R. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[4] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, page 16, 2008.
[5] . Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Demon: A local-first discovery method for overlapping communities. In *Proc. of KDD'12*, pages 615–623, New York, NY, USA, 2012. ACM.
[6] Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. In *Proc. of WWW'07*, pages 461–470. ACM, 2007.
[7] J. Eustace, X. Wang, and Y. Cui. Overlapping community detection using neighborhood ratio matrix. *Physica A: Statistical Mechanics and its Applications*, 421:510–521, 2015.
[8] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.
[9] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *PNAS'07*, 104(1):36–41, 2007.
[10] D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proc. of KDD'12*, pages 597–605. ACM, 2012.
[11] P. K. Gopalan and D. M. Blei. Efficient discovery of overlapping communities in massive networks. *PNAS'13*, 110(36):14534–14539, 2013.
[12] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In *Proc. SIGMOD'14*, pages 1311–1322. ACM, 2014.
[13] X. Huang, L. V. Lakshmanan, J. X. Yu, and H. Cheng. Approximate closest community search in networks. *Proc. of VLDB'15*, 9(4):276–287, 2015.
[14] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, May 2004.
[15] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.
[16] . Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *TCS*, 407(1):458–473, 2008.
[17] F. Luo, J. Z. Wang, and E. Promislow. Exploring local community structures in large networks. *Web Intelligence and Agent Systems*, 6(4):387–400, 2008.
[18] . Mihail, C. Gkantsidis, A. Saberi, and E. Zegura. On the semantics of internet topologies. Technical Report GIT-CC-02-07, College of Computing, Georgia Institute of Technology, Atlanta, GA, 2002.
[19] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.
[20] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77(1):016107, 2008.
[21] . E. Newman. Coauthorship networks and patterns of scientific collaboration. *PNAS'04*, 101(suppl 1):5200–5205, 2004.
[22] . E. Newman. Modularity and community structure in networks. *PNAS'06*, 103(23):8577–8582, 2006.
[23] . Rezvani, W. Liang, W. Xu, and C. Liu. Identifying top-k structural hole spanners in large-scale social networks. In *Proc. of CIKM'15*, pages 263–272, New York, NY, USA, 2015. ACM.
[24] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology*, pages 456–472. Springer, 2010.
[25] . Salathé and J. H. Jones. Dynamics and control of diseases in networks with community structure. *PLoS computational biology*, 6(4):e1000736, 2010.
[26] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proc. of WWW'15*, pages 927–937. ACM, 2015.
[27] J. Šíma and S. E. Schaeffer. On the NP-completeness of some graph cluster measures. In *SOFSEM 2006: Theory and Practice of Computer Science*, pages 530–537. Springer, 2006.
[28] C. Tsourakakis, J. Pachocki, and M. Mitzenmacher. Scalable motif-aware graph clustering. *arXiv preprint arXiv:1606.06235*, 2016.
[29] N. Wang, J. Zhang, K.-L. Tan, and A. K. Tung. On triangulation-based dense neighborhood graph discovery. *Proc. of VLDB'10*, 4(2):58–68, 2010.
[30] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *Proc. of CIKM'13*, pages 2099–2108, New York, NY, USA, 2013. ACM.
[31] Y. Wu, R. Jin, J. Li, and X. Zhang. Robust local community detection: On free rider effect and its elimination. *Proc. of VLDB'15*, 8(7):798–809, 2015.
[32] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Comput. Surv.*, 45(4):43:1–43:35, Aug. 2013.
[33] J. Xie, B. K. Szymanski, and X. Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *ICDMW'11*, pages 344–349. IEEE, 2011.
[34] W. Xu, M. Rezvani, W. Liang, J. X. Yu, and C. Liu. Efficient algorithms for the identification of top-$k$ strutural hole spanners in large social networks. *IEEE Transactions on Knowledge and Data Engineering*, 29(5):1–17–1030, 2017.
[35] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proc. of WSDM'13*, pages 587–596. ACM, 2013.
[36] Y. Zhang and S. Parthasarathy. Extracting analyzing and visualizing triangle k-core motifs within networks. In *ICDE'12*, pages 1049–1060. IEEE, 2012.

**Mojtaba Rezvani** received his MSc degree from Amirkabir University of Technology in Iran in 2012 and received his BSc from University of Qom in 2010, both in Computer Science. He is currently pursuing his PhD study in the Research School of Computer Science at the Australian National University. His research interests include graph databases, community detection, social networks, database, graph theory and algorithm design.

**Weifa Liang** (M'99-SM'01)received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in Computer Science. He is currently a Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of routing protocols for wireless ad hoc and sensor networks, cloud computing, graph databases, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE and a member of ACM.

**Chengfei Liu** received the BS, MS and PhD degrees in Computer Science from Nanjing University, China in 1983, 1985 and 1988, respectively. Currently he is a Professor in the Department of Computer Science and Software Engineering, Swinburne University of Technology. His current research interests include keyword search on structured data, graph database, social networks, query processing and refinement for advanced database applications, and data-centric workflows. He is a member of IEEE, and a member of ACM.

**Jeffery Xu Yu** has held teaching positions at the Institute of Information Sciences and Electronics, University of Tsukuba, and at the Department of Computer Science, Australian National University, Australia. Currently, he is a Professor in the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong, Hong Kong. His current research interests include graph data processing, graph mining, keyword search in relational databases, and social network analysis.