CCSS: Towards Conductance-based Community Search with Size Constraints

No Author Given

No Institute Given

Abstract. Size-constrained community search, retrieving a size-bounded high-quality subgraph containing user-specified query vertices, has been extensively studied in graph analysis. However, existing methods mainly focus on the cohesiveness within the community while ignoring its separation from the outside, leading to sub-optimal results. Inspired by this, we adopt the well-known *conductance* to measure the quality of a community and introduce a novel problem of *Conductance*-based Community Search with Size Constraints (*CCSS*). *CCSS* aims to identify a subgraph with the smallest *conductance* among all connected subgraphs that contain the query vertex q and have at least l and at most h vertices. We prove that *CCSS* is NP-hard. To efficiently address *CCSS*, we first propose a computational framework *ISC* with two-level loops. In the inner loop, *ISC* starts from q and iteratively adds vertices based on the predefined score function, obtaining a coarse-grained candidate community. In the outer loop, we design the perturbation strategy to repeatedly execute the inner loop until the quality of the candidate community converges. Then, we propose two heuristic algorithms, *ISCCI* and *ISCCP*, to concretely implement the score function and perturbation strategy of *ISC*, which significantly reduce the computational costs. Finally, empirical results demonstrate the superiorities of our solutions.

Keywords: Community Search · Size-constrainted Optimization Problem · Conductance.

1 Introduction

Community search, identifying a high-quality subgraph containing the user-given query vertices, has been widely studied in graph analysis [6, 8, 10, 13, 14, 18, 20, 23, 25]. Besides, community search also has numerous applications, such as impromptu activity organization [6], protein complexes identification [8], and social recommendation [20]. However, it is unreasonable to find a relatively small or large resultant community, as observed in [20, 26]. For example, consider a scenario where a project leader intends to organize a seminar with $5 \sim 9$ participants. The purpose of the seminar cannot be achieved if there are fewer than 5 people present, and the venue space cannot accommodate more than 9 people. Moreover, the absence of community size constraints can lead to diverse community sizes, complicating the result interpretation and application [17, 19, 26]. So, it is meaningful and interesting to impose size constraints on community search.

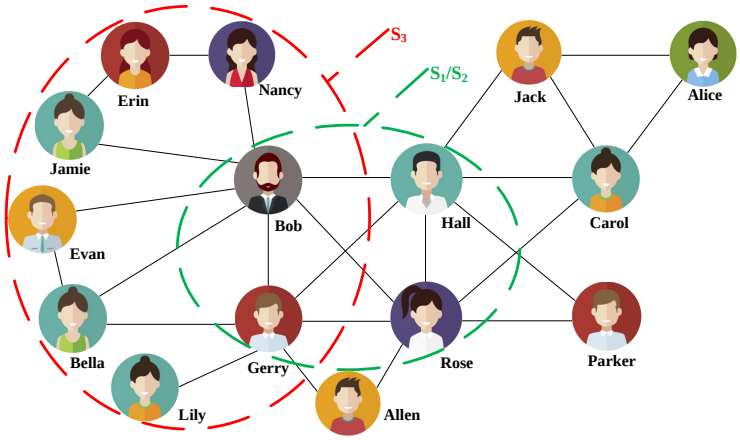


Fig. 1. Motivation Example

There are some studies that have been done on community search with size constraints (*CSS* for short) [17, 19, 20, 26]. Informally, *CSS* aims to identify a high-quality connected subgraph that contains the query vertex q and has at least l and at most h vertices. In the literature, Liu et al. [17] (resp., Yao et al. [26]) adopted the triangle-connected k-truss (resp., k-core) to model the *CSS* problem. Taking Figure 1 as an example, assume that *Bob* is the query vertex and $[l, h] = [4, 8]$. $S_1/S_2 = \{Bob, Greey, Hall, Rose\}$ is the resultant community returned by [17, 26]. However, almost all existing approaches use the cohesive subgraph to model *CSS*, which only focus on internal cohesiveness but ignores the separation from the outside of the output community, resulting in sub-optimal results [12, 24]. For example, the connections among *Bob*, *Greey*, *Hall*, and *Rose* are notably fewer than those connections outside of S_1/S_2, indicating evidently that S_1/S_2 is not an ideal community for *Bob*.

In this paper, we move the measure of community quality to *conductance*, which is to minimize the ratio of the number of edges going out of the community to the number of edges within the community [2]. Thus, *conductance* has strong interpretability because it takes simultaneously into account the internal and external connections of the community [16, 24]. Surprisingly, although *conductance* has been widely investigated in graph clustering [15, 16, 21], it has not yet been involved in *CSS* (Section 3). In Figure 1, the *conductance* of S_1/S_2 is 0.52, while the *conductance* of $S_3 = \{Evan, Bella, Bob, Nancy, Jamie, Erin, Lily, Gerry\}$ is 0.2, thus S_3 has better community separation. Inspired by these, we propose a novel problem of *Conductance*-based Community Search with Size constraints (named CCSS), which aims to identify the target community with the smallest *conductance* among all connected subgraphs that contain the user-specified query vertex q while satisfying the size constraints $[l, h]$.

Technical Challenges and Our Contributions. Existing methods either lack size constraints or fail to simultaneously consider both the internal and external connectivity of the community, rendering them ineffective in practical applica-

tions (Figure 1 and Section 5). However, identifying the exact *CCSS* raises significant challenges due to its NP-hardness (Theorem 1). Furthermore, we negatively uncover the *conductance* metric is not monotonic, thus the traditional greedy local search strategy cannot work [20]. To address these dilemmas, we introduce a novel clustering framework *ISC* with two-level loops. Specifically, in the inner loop, *ISC* starts from the query vertex q and iteratively adds qualified vertices into the candidate community based on the carefully design score function. During the above iteration, we maintain the subgraph with the smallest *conductance* among all connected subgraphs that satisfy the community size constraints. As a result, a coarse-grained candidate community is generated in the inner loop. In the outer loop, *ISC* adopts the perturbation strategy to repeatedly execute the inner loop until the quality of the candidate community converges. Namely, the outer loop can further improve the quality of the candidate community. Clearly, the primary computational bottleneck of *ISC* is how to design effective score functions and perturbation strategies to achieve a better trade-off between efficiency and accuracy. Thus, we first devise an efficient heuristic algorithm *ISCCI*, which employs the conductance increment as the vertex score function and the truncation perturbation to the current candidate community. To improve the quality, we propose another heuristic algorithm *ISCCP*, which uses the degree properties of vertices as the score function and the sum of the degree properties of the selected vertices in past iterations to perturb the current candidate community. In a nutshell, our main contributions are summarized as follows:

- **Novel Model.** To our knowledge, we are the first to use the well-known clustering model conductance to study the *CSS* problem, and propose a novel model of *Conductance*-based Community Search with Size Constraints (named *CCSS*). A striking feature of *CCSS* is that it can balance both external and internal connectivity within the community, while the previous models cannot.
- **Efficient Algorithms.** We develop a novel clustering framework *ISC* with two-level loops, which contains two key technologies: vertex score function and perturbation strategy. The vertex score function is to obtain a coarse-grained candidate community, while the perturbation strategy aims to refine the quality of the previous candidate community. Based on *ISC*, we design two heuristic algorithms *ISCCI* and *ISCCP* with near-linear time complexity.
- **Comprehensive Experiments.** Extensive experiments are conducted on eight datasets and five baseline competitors to thoroughly evaluate our solutions. The empirical results show that our proposal outperforms these baselines in terms of efficiency, scalability, and effectiveness.

2 PRELIMINARIES

2.1 Notations

Given an undirected and unweighted graph $G(V, E)$, we use $n = |V|$ (resp., $m = |E|$) to represent the number of vertices (resp., edges) of G. For an induced subgraph $S \subseteq G$ and the vertex $u \in S$, we let $N_S(u) = \{v \in S | (u, v) \in E\}$ be

the neighbors of u that belong to S. $d_S(u) = |N_S(u)|$ is the degrees of u in S. $|S|$ is the number of vertices of S. $\bar{S} = G - S$ is the complement of S.

Definition 1 (Edge Cut). *Given an induced subgraph $S \subseteq G$, the edge cut of S is $cut(S) = \{(u, v) \in E | u \in S, v \notin S\}$.*

Definition 2 (Vertex Volume). *Given an induced subgraph $S \subseteq G$, the vertex volume of S is $vol(S) = \sum_{u \in S} d_G(u)$.*

Definition 3 (Conductance [15, 16, 21]). *Given an induced subgraph $S \subseteq G$, the conductance of S is $\phi(S) = \frac{|cut(S)|}{min\{vol(S), 2m\text{-}vol(S)\}}$.*

As a consequence, a smaller $\phi(S)$ implies that the number of edges going out of S is relatively small compared with the number of edges within S. Namely, a smaller $\phi(S)$ means that S is densely connected internally and well separated from the remainder of G. Thus, we formulate our problem as follows.

Problem Definition. Given an undirected and unweighted graph $G(V, E)$, a query vertex $q \in V$, and two parameters l and h, Conductance-based Community Search with Size Constraints ($CCSS$) aims to identify a subgraph S, satisfying

(1) Connectivity: S is connected and contains q;

(2) Size Constraint: S satisfies the size constraint, i.e., $l \leq |S| \leq h$;

(3) Optimal Conductance: $\phi(S)$ is minimized among all subgraphs of G satisfying the above two conditions.

The subgraph that meets the connectivity and size constraint is called a candidate community, and the candidate community that meets the optimal conductance condition is called the target community.

Example 1. Reconsider Figure 1, we assume that query vertex $q = Bob$ and the size constraint $[l, h] = [4, 8]$. According to the Problem Definition, one of the candidate communities is $S_1 = \{Bob, Greey, Hall, Rose\}$ and $\phi(S_1) = 0.52$. $S_3 = \{Evan, Bella, Bob, Nancy, Jamie, Erin, Lily, Gerry\}$ is the target community with the smallest *conductance* $\phi(S_3) = 0.2$.

Theorem 1. *Given an undirected and unweighted graph $G(V, E)$, a query vertex $q \in V$, and two parameters l and h, identifying the CCSS from G is NP-hard.*

All proofs are deferred to the Technical Report [1] due to the space limit.

3 Existing Solutions and Our Basic Solution

3.1 Existing Solutions

Conductance-based graph clustering methods [2, 15, 16, 21] have many similarities with our problem, but there are also pivotal differences (see Table 1 for details). Specifically, *conductance*-based graph clustering methods first compute a score function of each vertex by graph diffusions (e.g., Personalized PageRank [2], Heat Kernel PageRank [15]) or eigenvector [21] or degree ratio [16].

Table 1. A comparison of conductance-based graph clustering algorithms. ϵ is the error tolerance. α and t are model parameters of *NIBBLE_ PPR* and *HK_ Relax*, respectively. $\times$ represents the corresponding method has not applicable to the *CSS* problem.

Methods	Time Complexity	Space Complexity	CSS	Remark
PCon_ de [16]	$O(n+m)$	$O(n+m)$	$\times$	Degree Ratio-based
ASC [21]	$O((n+m)\frac{1}{\epsilon}log\frac{n}{\epsilon})$	$O(n+m)$	$\times$	Eigenvector-based
NIBBLE_ PPR [2]	$O(\frac{log\frac{1}{\alpha\epsilon}}{\alpha\epsilon})$	$O(n+m)$	$\times$	Diffusion-based
HK_ Relax [15]	$O(\frac{te^t log(\frac{1}{\epsilon})}{\epsilon})$	$O(n+m)$	$\times$	Diffusion-based
ISCCI (this paper)	$O(nlogn)$	$O(n+m)$	$\surd$	Diffusion-based
ISCCP (this paper)	$O(nlogn)$	$O(n+m)$	$\surd$	Diffusion-based

Then, they iteratively add the vertex with the largest score. Finally, they output the result with the smallest *conductance* during the previous iteration. As a consequence, these existing methods may not satisfy the connectivity and size constraint of our proposed *CCSS*. Namely, the communities they identify are disconnected or do not contain the given query vertex or even have unreasonable size (see Section 5 for details). In particular, [9, 28] have proved both theoretically and empirically that existing *conductance*-based graph clustering methods violate the connectivity condition of our proposed *CCSS*.

3.2 Our Basic Solution: CS_PPR

In this section, we propose a basic heuristic solution *CS_ PPR* (i.e., Algorithm 1) by adjusting the existing *conductance*-based graph clustering methods. Specifically, Algorithm 1 first obtains a probability distribution $\hat{\pi}$ by executing the *Forward_ Push* subroutine proposed by Anderson et. al [2] (Line 1). Subsequently, it changes $\hat{\pi}(q)$ to infinity to ensure that q is included in the resultant community S (Line 2). Finally, Lines 3-9 execute a sweep cut procedure over the vector y to obtain the local optimal connected subgraph S containing q.

Disadvantages of *CS_ PPR*. Although *CS_ PPR* can obtain a candidate community S (i.e., S meets the connectivity and size constraint of our problem), it is still inefficient and ineffective in practice (Section 5). In particular, *CS_ PPR* is heuristic and the quality of its output results is heavily dependent on many hard-to-tune parameters, resulting in their performance being unstable and in most cases very poor. For example, the larger α or r_{max} is, the faster it convergences, potentially reducing the quality of the results. Conversely, the smaller α or r_{max} is, the more computationally expensive it is. As a consequence, how to choose the proper parameters is a major challenge for users.

4 Our Advanced Solutions

4.1 Overall Algorithm Framework: ISC

As stated in Section 3.1, [2, 15, 16, 21] can be summarized as a three-stage algorithm. However, a single round of such a search may fall into a local optimum.

Algorithm 1 *CS_PPR(G, q, [l,h], α, r_{max})*

Input: An undirected graph G; query vertex q; size bonded $[l, h]$; jump probability α and termination threshold r_{max}
Output: A community S

$\hat{\pi} \leftarrow$ FORWARD_PUSH(G, q, α, r_{max}) [2]
$\hat{\pi}(q) \leftarrow \infty$; $y \leftarrow \hat{\pi} D^{-1}$
$y_i \leftarrow$ the index of y with ith largest value
$S \leftarrow \emptyset$; $\phi^* \leftarrow \infty$
for $i = l$ to h **do**
    $S_i \leftarrow \{y_1, y_2, ...y_i\}$
    if S_i is connected and $\phi(S_i) < \phi^*$ **then**
        $S \leftarrow S_i$; $\phi^* \leftarrow \phi(S_i)$
return S
function FORWARD_PUSH(G, q, α, r_{max})
    $\hat{\pi}(t) \leftarrow 0$ for all $t \in V$
    $r(q) \leftarrow 1$; $r(t) \leftarrow 0$ for all $t \neq q$
    while $\exists t$ such that $r(t) \geq r_{max} \cdot d_G(t)$ **do**
        for each $u \in N_G(t)$ **do**
            $r(u) \leftarrow r(u) + (1 - \alpha) r(t)/d_G(t)$
        $\hat{\pi}(t) \leftarrow \hat{\pi}(t) + \alpha r(t)$, $r(t) \leftarrow 0$
    return $\hat{\pi}$

Therefore, we introduce a four-stage bottom-up iterative search clustering framework *ISC* based on the original framework. In the first stage, we assign a predefined $score(u)$ to the vertex $u \in N_{\bar{S}}(S)$. In the second stage, we iteratively add u with $score(u)_{max}$ to enhance the clustering and expand the search scope. In the third stage, the community with the smallest *conductance* is selected as the target community for the first iteration. Finally, we use a certain perturbation strategy for multiple iterations to optimize and output the target community.

Obviously, *ISC* continuously searches and improves within a small range around the current solution. This gradual refinement process enables *ISC* to make progress towards the optimal solution while steering clear of local minima. The focus of *ISC* is on stage 1 and 4. Therefore, in the following algorithms, we design vertex scores and perturbation strategies to guide the selection of vertices.

Theorem 2. *The time complexity and the space complexity of* ISC *are* $O(nlogn)$ *and* $(n + m)$*, respectively.*

4.2 The *ISCCI* Algorithm

A local community is said to be "strengthened" when its *conductance* decreases. We give several important theorems as follows.

Theorem 3 (Nonmonotonicity). *Given an induced subgraph* $S \subseteq G$*, for any neighbor* $v_i \in N_{\bar{S}}(S)$*,* $\phi(S \cup \{v_i\})$ *may increase, decrease or remain unchanged.*

Example 2. Consider Figure 2, the initial community $S = \{v_0, v_5\}$ with $\phi(S) = 5/7$. To expand S, we have the option to add v_6 or other neighbors to S. If we choose to include v_1 in S, *conductance* dramatically drops to $1/2$. Conversely, incorporating v_6 into S results in an increase in community *conductance* of 1.

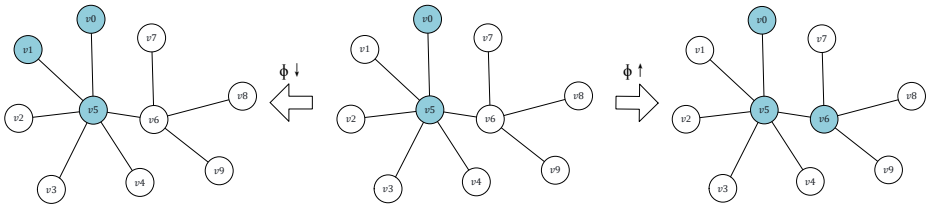


Fig. 2. Running example to illustrate the nonmonotonicity of *conductance*.

Algorithm 2 *ISCCI(G, q, [l,h], T)*

Input: An undirected graph G; iteration count T; query vertex q; size bonded $[l, h]$
Output: A cohesive and exo-sparse graph $\hat{S}$

$S_1 \leftarrow \{q\}$; $\hat{S} \leftarrow None$; $\hat{\phi}(S) \leftarrow INF$
for $t : 1 \rightarrow T$ **do**
  if $t \geq 2$ **then**
    Randomly select a breakpoint $w \in \hat{S} \backslash \{q\}$ and divide it into $(S' \backslash \{w\}, S'' \cup \{w\})$
    $S_{t+1} \leftarrow S' \backslash \{w\}$
  while $|S_t| < h$ and $N_{\bar{S}_t}(S_t) \neq \emptyset$ **do**
    Find the vertex $u \in N_{\bar{S}_t}(S_t)$ with the maximum $\triangle c(u)$ and $u \neq$ breakpoint
    $S_t \leftarrow S_t \cup \{u\}$
    if $|S_t| \in [l, h]$ and $\phi(S_t) \leq \hat{\phi}(S)$ **then**
      $\hat{S} \leftarrow S_t$; $\hat{\phi}(S) \leftarrow \phi(S_t)$
      for $v_i \in N_{\bar{S}_t}(S_t)$ **do**
        update $\triangle c(v_i)$ by Theorem 5
Return $\hat{S}$

By adding v_i in order, we can get a solution sequence $S_i = \{q, v_1, \cdots, v_i\}$. Different addition orders of v_i will lead to different S_i, making the change of $\phi(S_{i-1} \cup \{v_i\})$ non-monotonic during the addition process. Surprisingly, we find that there are sequences of non-increasing functions $\phi(S_i)$ of v_i such that the *conductance* changes monotonically.

Theorem 4. *Given an induced subgraph $S \subseteq G$, for any neighbor $v_i \in N_{\bar{S}}(S)$, there always exists a vertex sequence $\{q, v_1, \cdots, v_i\}$ starting with the query vertex $q \in S$ such that $\phi(S) \geq \phi(S_1) \geq, \cdots, \geq \phi(S_i)$.*

Theorem 5 (Conductance Increment). *Given an induced subgraph $S \subseteq G$ and adding any neighbor $v \in N_{\bar{S}}(S)$, the conductance increment is expressed as $\triangle c = \phi(S) - \phi(S \cup \{v\}) = \frac{cut(S)d_G(v) - min\{vol(S), 2m\text{-}vol(S)\}(d_G(v) - 2d_S(v))}{min\{vol(S), 2m\text{-}vol(S)\}(min\{vol(S), 2m\text{-}vol(S)\} + d_G(v))}$, where $cut(S) = \sum_{u_i \in S}(d_G(u_i) - 2d_S(u_i)), vol(S) = \sum_{u_i \in S} d_G(u_i)$.*

We use $\triangle c$ as the vertex score. The bigger $\triangle c$ is, $S \cup \{v\}$ is "strengthened" more obviously. Selecting the vertices with $\triangle c_{max}$ at each step can generate a unique vertex sequence $\hat{S}$. Then, a truncation strategy is adopted for the next iteration. Specifically, randomly intercept the part of $\hat{S}$ that contains the query vertex q and excludes the breakpoint, and perform a greedy search again.

We propose an *ISCCI* algorithm that combines conductance increment $\triangle c$ and truncation perturbation, and the pseudocode is shown in Algorithm 2. In

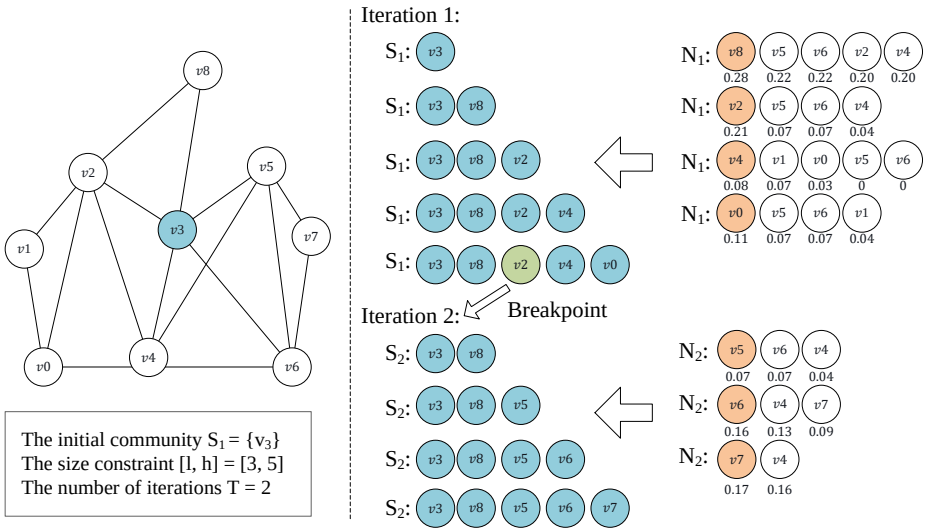


Fig. 3. Running example of *ISCCI* based on $\triangle c$ and truncated perturbation.

each iteration, we greedily merge the vertex u with $\triangle c(u)_{max}$ into the community S_t (Lines 7-8). When S_t meets the conditions of the candidate community, we determine whether S_t needs to be updated by comparing *conductance* (Lines 9-10), and update $\triangle c$ of $N_{\bar{S}_t}(S_t)$ through Theorem 5 (Lines 11-12). Until all candidate communities have been explored, we randomly select a breakpoint w from $\hat{S}$ and return the segment S' that contains q and excludes w as the initial solution for the next iteration (Lines 4-5). After T iterations, we output the community $\hat{S}$ with the smallest *conductance* $\hat{\phi}(S)$ as the result (Line 13).

Example 3. Figure 3 shows a running example and operation steps of Algorithm 2. According to Algorithm 2, the conductance increments of $N_{\bar{S}_1}(v_3)$ are $\{v_8 : 0.28, v_5 : 0.22, v_6 : 0.22, v_2 : 0.20, v_4 : 0.20\}$, so we choose v_8 to S_1. Then, we update $\triangle c$ of $N_{\bar{S}_1}(S_1)$, and add v_2, v_4, and v_0 to S_1 according to the same selection strategy. The optimal community after the first iteration is $\hat{S} = \{v_3, v_8, v_2\}$ with $\hat{\phi}(S)_{min} = 0.5$. Next, we arbitrarily intercept a part of $\hat{S}$ containing $v_3 : S_2 = \{v_3, v_8\}$, and $\triangle c$ of $N_{\bar{S}_2}(S_2)$ are $\{v_5 : 0.07, v_6 : 0.07, v_4 : 0.04\}$. Continuously select the vertex with $\triangle c_{max}$ and update $\triangle c$ of $N_{\bar{S}_2}(S_2)$. v_5, v_6 and v_7 are added to S_2 in turn. Finally, through two rounds of iterations, we obtain the target community $\hat{S} = \{v_3, v_8, v_5, v_6\}$ with $\hat{\phi}(S)_{min} = 0.467$.

4.3 The *ISCCP* Algorithm

As shown in Algorithm 2, each addition of a vertex will cause the *conductance* to change, which requires updating $\triangle c$ of all neighbor vertices. To update only the neighbors of the selected vertex, we consider the properties of the vertex itself.

Algorithm 3 *ISCCP(G, q, [l,h], T)*

Input: An undirected graph G; iteration count T; query vertex q; size bonded $[l, h]$
Output: A cohesive and exo-sparse graph $\hat{S}$

1: $S_1 \leftarrow \{q\}$; $\hat{S} \leftarrow None$; $\hat{\phi}(S) \leftarrow INF$
2: Initialize all vertices $f^0(u) \leftarrow 0$; update $f^1(u \in N_{\bar{S_1}}(S_1))$ by Theorem 6
3: **for** $t : 1 \rightarrow T$ **do**
4: **while** $|S_t| < h$ and $N_{\bar{S_t}}(S_t) \neq \emptyset$ **do**
5: Find the vertex $u \in N_{\bar{S_t}}(S_t)$ with the minimum load $f^t(u)$
6: $S_t \leftarrow S_t \cup \{u\}$
7: **if** $|S_t| \in [l, h]$ and $\phi(S_t) \leq \hat{\phi}(S)$ **then**
8: $\hat{S} \leftarrow S_t$; $\hat{\phi}(S) \leftarrow \phi(S_t)$
9: **for** $v_i \in N_{\bar{S_t}}(u)$ **do**
10: update $f^{t+1}(v_i)$ by Theorem 7
11: $S_{t+1} \leftarrow \{q\}$
12: Return $\hat{S}$

Inspired by the ideal community model of strong internal cohesion and weak external connections, we notice that high-quality communities tend to exhibit vertices with stronger internal connections than external connections.

Theorem 6 (Degree Property). *Given an induced subgraph $S \subseteq G$, the exo-sparse and cohesion property of a single vertex $u \in N_{\bar{S}}(S)$ is $g(u) = \frac{d_G(u)}{d_S(u)}$.*

According to Theorem 6, a unique vertex sequence can be obtained by selecting vertex of g_{min} each time. However, when the selection order of u is changed only once, $g(v)$ of $v \in N_{\bar{S}}(u)$ will be affected, and thus a different vertex selection sequence may be obtained. To obtain more vertex sequences and increase the possibility of other vertices being selected in the next iteration, we comprehensively consider the load of the vertices as a perturbation in Theorem 7.

Theorem 7 (Vertex Load). *For any vertex $u \in N_{\bar{S}}(S)$ and the number of iterations T, the vertex load is $f^t(u) = \sum_{t \in T} g^t(u)$.*

The pseudocode of the *ISCCP* algorithm based on vertex load is shown in Algorithm 3. In each iteration, the vertex u with $f^t(u)_{min}$ is added to the community S_t (Lines 5-6). It should be noted that if u is not selected in the t iteration, then $g^t(u) = 0$. The candidate community with the smallest *conductance* and the loads of $N_{\bar{S}}(u)$ are updated each time (Lines 7-10). If the scale of S_t exceeds h, we stop the internal search and perform the next weighted iterative search (Line 11). Until T iterations are completed, the community with the smallest *conductance* is returned as the target community $\hat{S}$ (Line 12).

Example 4. Consider the same setting as Example 3, Figure 4 illustrates the steps of Algorithm 3. In the first iteration, the loads of $N_{\bar{S_1}}(v_3)$ are $\{v_8 : 2, v_5 : 4, v_6 : 4, v_2 : 5, v_4 : 5\}$ and we add v_8 to S_1 while updating its new neighbor load $f(v_2) = 2.5$. v_2, v_1 and v_0 are added to S_1 in turn in the same way. After the first round of iteration, the current target community is $\{v_3, v_8, v_2, v_1\}$ with $\phi_{min} =$

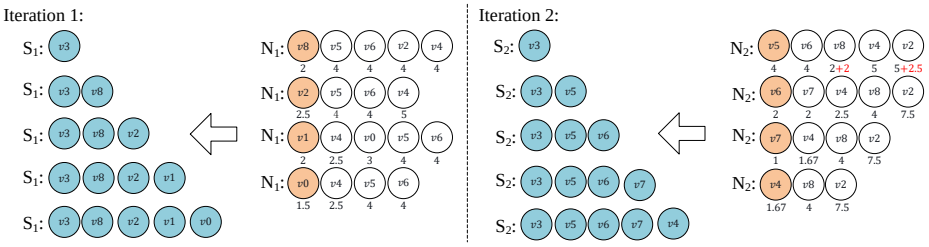


Fig. 4. Running example of *ISCCP* based on vertex load information.

0.428. Then proceed to the second round of iteration, $S_2 = \{v_3, v_5\}$ because $f(v_2) = 5 + 2.5 = 7.5$, $f(v_4) = 5 + 0 = 5$, $f(v_5) = 4 + 0 = 4$, $f(v_6) = 4 + 0 = 4$, $f(v_8) = 2 + 2 = 4$. Similarly, v_6, v_7 and v_4 are added to S_2 in order. Finally, the target community $\hat{S} = \{v_3, v_5, v_6, v_7\}$ with $\phi(\hat{S})_{min} = 0.333$ is returned.

5 Experimental Evaluation

5.1 Experimental Setup

Datasets and Algorithms. Our solutions are evaluated on real-life and synthetic graphs (Table 2), which are widely used benchmarks for community search [17, 19, 26, 16]. The first six graphs are obtained from publicly-available datasets[1] and the last two are synthetic datasets with 1,000,000 vertices [3, 22]. *CSM* [8], *HK-Relax* [15], *SC-BRB* [26], *STCS* [27], and *CS_PPR* are chosen for comparison with our proposed *ISCCI* and *ISCCP*. These algorithms are conducted on a Linux server with an Intel(R) Xeon(R) E5-2683 v3@2.00GHZ CPU and 256GB RAM running CentOS 6.10. Detailed descriptions of datasets and algorithms are deferred to the Technical Report [1] due to the space limit.

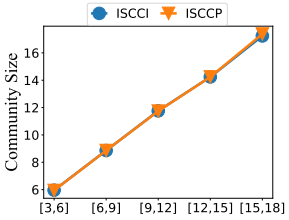
Effectiveness Metrics. *Conductance* (*Con* for short) [15, 16] and *Subgraph Modularity* (*SM* for short) [18, 5, 11, 23] are chosen to evaluate the quality of the identified community. Specifically, *Con* is calculated by Definition 3 and $SM = \sum_{u \in S} d_S(u) / \sum_{u \in S} d_{\bar{S}}(u)$. The smaller the value of the former, the better the partition of community S, and vice versa for the latter.

Parameters and Query Vertices. Consistent with [26], the size of the community is limited to $[l, h] \in \{[3, 6], [6, 9], [9, 12], [12, 15], [15, 18]\}$. For parameters related to *HK-Relax* and *CS_PPR*, we default $\epsilon = 1/m$, $t = 5$, and $\alpha = 0.01$. The number of iterations of our algorithms is $T = 1000$. Each time, 50 query vertices with degree greater than the average degree are randomly selected.

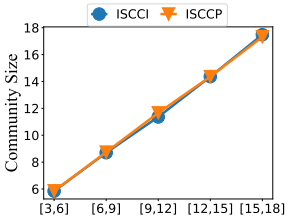
[1] These datasets are available from http://snap.stanford.edu/

Table 2. Dataset statistics. d_{avg} is the average degree and d_{max} is the maximum degree.

Dataset	Type	n	m	d_{avg}	d_{max}
Amazon[1]	Social	334,863	925,872	5.52	549
DBLP[1]	Authorship	317,080	1,049,866	6.62	343
Youtube[1]	Social	1,134,890	2,987,624	5.26	28.754
Google[1]	Web	855,802	4,291,352	10.02	6,332
Pokec[1]	Social	1,632,803	30,622,564	27.31	14,854
LiveJ[1]	Social	3,997,962	34,681,189	17.34	14,815
BA [3]	Synthetic	1,000,000	5,999,964	11.99	4,265
WS [22]	Synthetic	1,000,000	7,000,000	14	21

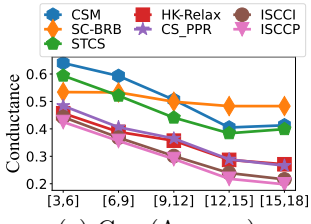


(a) Amazon

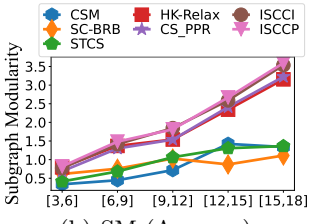


(b) DBLP

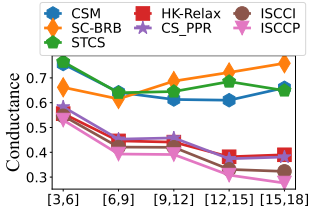
Fig. 5. Result size



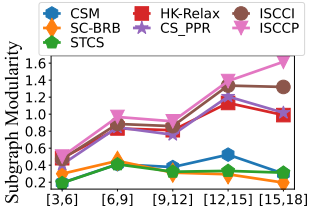
(a) Con (Amazon)



(b) SM (Amazon)



(c) Con (DBLP)



(d) SM (DBLP)

Fig. 6. Result quality

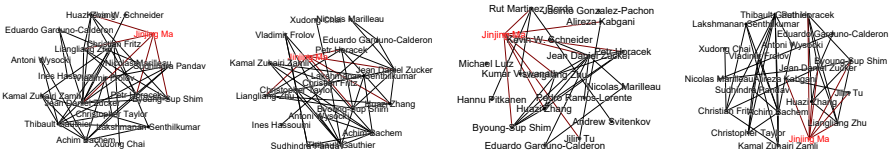
5.2 Effectiveness Testing

Exp-1: Result Size. Figure 5 shows the community sizes returned by *ISCCI* and *ISCCP*. The x-axis represents the constraint range from $[3, 6]$ to $[15, 18]$ and the y-axis represents the average size of the identified communities. The community sizes returned by our algorithms are both within the given constraints.

Exp-2: Result Quality. Figure 6 shows the *Con* and *SM* of the algorithms under different $[l, h]$ on Amazon and DBLP. Because *CSM*, *SC-BRB*, and *STCS* focus solely on enhancing internal cohesion within the community, they result in inferior *Con* and *SM*. In contrast, the remaining algorithms consider both tight internal connections and sparse external ones. However, our algorithms achieve lower *Con* and larger *SM* thanks to the advantage of the iterative local clustering framework in balancing the optimal solution. We observe that the trends in Table 3 are similar to those in Figure 6. Specifically, compared with

Table 3. *Con* and *SM* on datasets under the size constraint $[l, h] = [15, 18]$. The best and second-best results in each metric are marked in **bold** and underlined, respectively.

Model	Amazon		DBLP		Youtube		Google		Pokec		LiveJ		BA		WS	
	Con	SM	Con	SM	Con	SM	Con	SM	Con	SM	Con	SM	Con	SM	Con	SM
CSM	0.41	1.33	0.66	0.30	0.93	0.05	0.78	0.27	0.90	0.05	0.79	0.18	0.93	0.03	0.36	0.90
SC-BRB	0.48	1.11	0.75	0.19	0.99	0.00	0.78	0.24	0.91	0.04	0.84	0.15	0.99	0.00	0.32	1.07
STCS	0.39	1.36	0.64	0.31	0.98	0.00	0.78	0.28	0.8	0.06	0.80	0.16	0.99	0.00	0.35	0.99
HK-Relax	0.27	3.15	0.39	0.98	0.51	0.70	0.48	0.83	0.83	0.10	0.68	0.30	0.79	0.12	0.29	1.22
CS_PPR	0.26	3.22	0.38	1.01	0.51	0.74	0.50	0.77	0.85	0.08	0.70	0.25	0.82	0.10	0.29	1.21
ISCCI	0.21	3.53	0.32	1.32	0.44	0.93	0.42	0.93	0.75	0.16	0.59	0.43	0.74	0.17	0.28	1.25
ISCCP	**0.19**	**3.59**	**0.27**	**1.61**	**0.41**	**1.00**	**0.36**	**1.06**	**0.68**	**0.23**	**0.53**	**0.50**	**0.74**	**0.16**	**0.28**	**1.27**



(a) *ISCCI* (b) *ISCCP* (c) *HK-Relax* (d) *CS_PPR*

Fig. 7. A CSS case study on DBLP.

CS_PPR, the *Con* of our best algorithm is improved by about 0.7˜0.9 times, and the *SM* is improved by $1.1 \sim 2.8$ times.

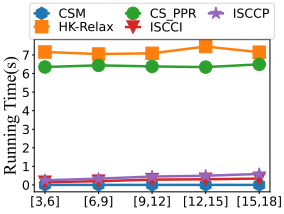
Exp-3: A Case Study on DBLP. We perform a *CSS* case study on DBLP to identify associates of the author "$JinjingMa$" within the range of $[15, 20]$. Figure 7 confirms communities within the desired size range. However, the community returned by *HK-Relax* is disconnected. Furthermore, our algorithms achieve higher clustering coefficients of 1 for the query vertices, while *HK-Relax* and *CS_PPR* have clustering coefficients of 0.392 and 0.523, respectively. When computing *conductance*, *ISCCI* and *ISCCP* record values of 0.2 and 0.19, while *HK-Relax* and *CS_PPR* report *conductance* values of 0.49 and 0.26, respectively.

5.3 Efficiency Testing

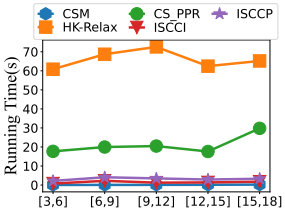
Exp-4: Different Algorithms for Comparison. Figure 8 shows the running time of the algorithms under different $[l, h]$ on DBLP and Youtube. *CSM* has the fastest runtime because it finds subgraphs with the maximum k, which may lead to early termination and communities smaller than l. Excluding *CSM* from further tests ensures a fair comparison. Unlike our algorithms, *CS_PPR* and `HK-Relax` need to calculate the probability distribution of vertices in advance, involving unnecessary vertices that are more than h away from the query vertex, which can be very time-consuming. In contrast, our algorithms focus on neighbor vertices within size constraints in each step, ensuring high efficiency. Table 4 mirrors the trend seen in Figure 8, providing a comparison of algorithm running times across all datasets in the $[15, 18]$ range. Notably, our algorithms are several to hundreds of times faster than *CS_PPR*.

Table 4. Running time (seconds) of various methods under the size constraint $[l, h] = [15, 18]$. AVG.RANK is the average rank of each method across testing datasets.

Model	Aamazon	DBLP	Youtube	Google	Pokec	LiveJ	BA	WS	AVG.RANK
HK-Relax	0.61	7.14	65.17	6.09	816.70	793.33	137.56	10.41	4
CS_PPR	8.69	6.49	29.78	21.12	143.89	511.72	8.06	21.26	3
ISCCI	**0.25**	**0.34**	**1.75**	**1.63**	**3.00**	**2.44**	**0.94**	**0.53**	**1**
ISCCP	0.42	0.59	3.30	2.87	3.67	3.69	1.29	1.07	2

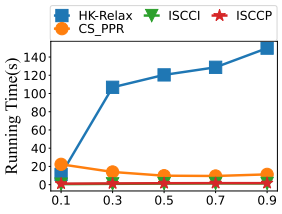


(a) DBLP



(b) Youtube

Fig. 8. Running time of various algorithms under different constraints $[l, h]$.



(a) Google

Fig. 9. Running time (vary p)

Exp-5: Different Reconnection Probabilities p in WS. Figure 9 shows increasing the rewiring probability p of WS from 0.1 to 0.9 at 0.2 intervals to evaluate the running time of the algorithms under $[15, 18]$. As the graphs grow in complexity due to increased p, necessitating consideration of more vertices, the running time of *HK-Relax* notably increases, while the remaining algorithms have smaller increments. Throughout, *ISCCI* and *ISCCP* consistently exhibit shorter running times compared to *CS_PPR* and *HK-Relax*.

Exp-6: Different Parameters of Heuristics. Figure 10 (a-b) shows that larger t values lead to slower convergence in *HK-Relax* as the heat kernel assigns more weight to distant vertices during the walk. Similarly, in Figure 10 (c-d), increasing α enhances the efficiency of *CS_PPR* by facilitating more frequent transitions to neighboring vertices, speeding up convergence to the subgraph around the seed. Figure 10 (e-f) illustrates that setting ϵ to a very small value slows down both *HK-Relax* and *CS_PPR* as they require repeated vector difference checks, even when they are close to convergence. These comparisons highlight the faster and more stable running time of our algorithms.

Exp-7: Different Lower or Upper Bounds. Figure 11 (a-b) shows that when l is held constant, time consumption rises with an increasing h due to the enlarged search space involving more vertices, leading to greater time overhead. Conversely, Figure 11 (c-d) shows that with a fixed h, the runtime of $ISCCI$ decreases as l grows, while the runtime of $ISCCP$ remains relatively stable. This distinction arises because $ISCCI$ updates conductance increments for all $N_{\bar{S}}(S)$ at each step, while $ISCCP$ only updates the load of $N_{\bar{S}}(u)$.

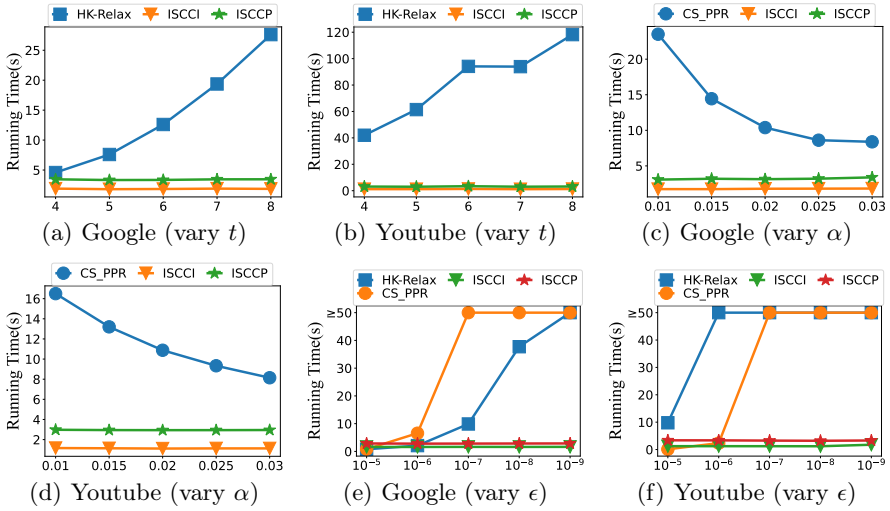


Fig. 10. Running time of various algorithms under different parameters.

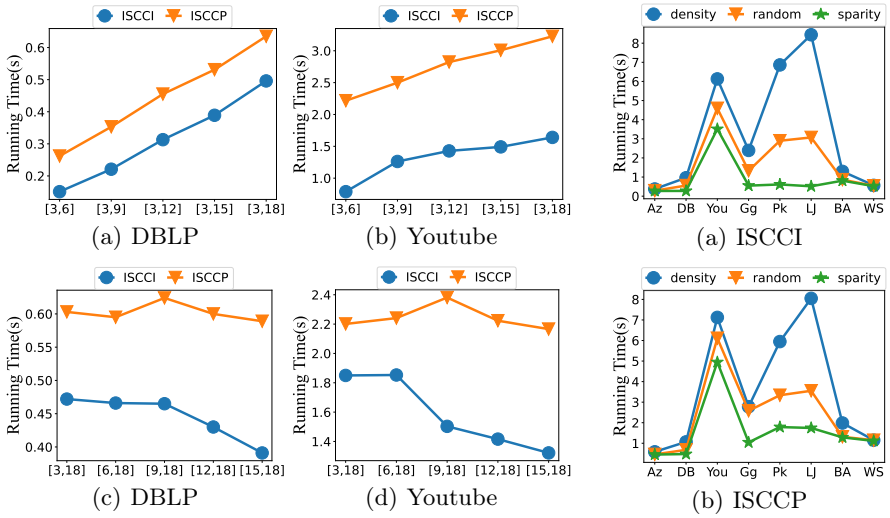


Fig. 11. Running time of different lower or upper bounds (vary $[3, h]\&[l, 18]$).

Fig. 12. Running time of different query vertex sets.

Exp-8: Different Query Vertex Sets. Figure 12 shows runtime for different vertex query sets under size constraints $[15, 18]$. Queries are categorized into three types based on degree from dense to sparse: dense (top 5%), sparse (bottom 5%), and mixed (randomly selected). Sparse query sets lead to faster community localization due to fewer interconnecting edges, reducing conductance increment

or load computation time. In contrast, dense query sets exhibit slower query efficiency, while mixed query sets, being a mix of the two, fall in between.

6 Other Related Work

The problem of community search can be broadly divided into size-bounded community search [13, 7, 4, 8, 20, 23] and size-unbounded community search [17, 19, 20, 26, 27]. The former aims to identify the subgraphs that contain the given query vertices and satisfy a specific community model such as k-core [20, 8], k-truss [13], clique [7] and density [23]. However, the absence of community size constraints may result in identified communities that are too large or too small, making it difficult to interpret the community structure [20, 27]. To alleviate these defects, the latter aims to find the community under size constraints, which is more personalized and utility. For instance, both Sozio et al. [20] and Ma et al. [19] adopted k-core to search a connected community with at most h vertices. Since the k-core is not necessarily cohesiveness, Zhang et al. [27] focused on applying the higher-order k-truss to find the maximum triangular connectivity subgraph while satisfying the size constraints $[l, h]$. Unfortunately, these existing approaches ignore the separation from the outside of the community, leading to sub-optimal results. Thus, they cannot be directly extend to solve our problem.

7 Conclusion

In this paper, we are the first to study the problem of conductance-based community search with size constraints, aiming to identify a connected subgraph that contains the user-initiated query vertex and has the smallest conductance under size constraints. We proved that our problem is NP-hard. Thus, we first developed a two-level iterative search framework *ISC* to heuristically address our problem. Building upon *ISC*, we then proposed two algorithms, *ISCCI* and *ISCCP*, which not only consider both the score function and perturbation strategy of *ISC* but also avoid unnecessary calculations. Finally, extensive experiments on real-world and synthetic datasets confirmed that our solutions are more efficient, scalable, and effective than the state-of-the-art competitors.

References

1. Technical report for ccss: Towards conductance-based community search with size constraint. https://github.com/Lin021/QTCS
2. Andersen, R., Chung, F.R.K., Lang, K.J.: Local graph partitioning using pagerank vectors. In: FOCS. pp. 475–486 (2006)
3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science **286**(5439), 509–512 (1999)
4. Barbieri, N., Bonchi, F., Galimberti, E., Gullo, F.: Efficient and effective community search. Data Min. Knowl. Discov. **29**(5), 1406–1433 (2015)
5. Chen, J., Zaïane, O.R., Goebel, R.: Local community identification in social networks. In: Memon, N., Alhajj, R. (eds.) ASONAM. pp. 237–242 (2009)

6. Chen, L., Liu, C., Zhou, R., Xu, J., Yu, J.X., Li, J.: Finding effective geo-social group for impromptu activities with diverse demands. In: KDD. pp. 698–708 (2020)
7. Cui, W., Xiao, Y., Wang, H., Lu, Y., Wang, W.: Online search of overlapping communities. In: SIGMOD. pp. 277–288 (2013)
8. Cui, W., Xiao, Y., Wang, H., Wang, W.: Local search of communities in large graphs. In: SIGMOD. pp. 991–1002 (2014)
9. Ding, X., Zhang, J., Yang, J.: A robust two-stage algorithm for local community detection. Knowl. Based Syst. **152**, 188–199 (2018)
10. Fang, Y., Huang, X., Qin, L., Zhang, Y., Zhang, W., Cheng, R., Lin, X.: A survey of community search over big graphs. VLDB J. **29**(1), 353–392 (2020)
11. Fanrong, M., Mu, Z., Yong, Z., Ranran, Z., et al.: Local community detection in complex networks based on maximum cliques extension. Mathematical Problems in Engineering **2014** (2014)
12. Fortunato, S.: Community detection in graphs. Physics Reports **486**(3), 75–174 (2009)
13. Huang, X., Cheng, H., Qin, L., Tian, W., Yu, J.X.: Querying k-truss community in large and dynamic graphs. In: SIGMOD. pp. 1311–1322 (2014)
14. Huang, X., Lakshmanan, L.V.S., Yu, J.X., Cheng, H.: Approximate closest community search in networks. PVLDB **9**(4), 276–287 (2015)
15. Kloster, K., Gleich, D.F.: Heat kernel based community detection. In: Macskassy, S.A., Perlich, C., Leskovec, J., Wang, W., Ghani, R. (eds.) KDD. pp. 1386–1395 (2014)
16. Lin, L., Li, R., Jia, T.: Scalable and effective conductance-based graph clustering. In: AAAI (2023)
17. Liu, B., Zhang, F., Zhang, W., Lin, X., Zhang, Y.: Efficient community search with size constraint. In: ICDE. pp. 97–108 (2021)
18. Luo, F., Wang, J.Z., Promislow, E.: Exploring local community structures in large networks. In: IEEE. pp. 233–239 (2006)
19. Ma, Y., Yuan, Y., Zhu, F., Wang, G., Xiao, J., Wang, J.: Who should be invited to my party: A size-constrained k-core problem in social networks. J. Comput. Sci. Technol. **34**(1), 170–184 (2019)
20. Sozio, M., Gionis, A.: The community-search problem and how to plan a successful cocktail party. In: KDD. pp. 939–948 (2010)
21. Trevisan, L.: Lecture notes on graph partitioning, expanders and spectral methods. University of California, Berkeley, https://people. eecs. berkeley. edu/luca/books/expanders-2016. pdf (2017)
22. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. nature **393**(6684), 440–442 (1998)
23. Wu, Y., Jin, R., Li, J., Zhang, X.: Robust local community detection: On free rider effect and its elimination. Proc. VLDB Endow. **8**(7), 798–809 (2015)
24. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowl. Inf. Syst. **42**(1), 181–213 (2015)
25. Yang, R., Xiao, X., Wei, Z., Bhowmick, S.S., Zhao, J., Li, R.: Efficient estimation of heat kernel pagerank for local clustering. In: SIGMOD. pp. 1339–1356 (2019)
26. Yao, K., Chang, L.: Efficient size-bounded community search over large networks. Proc. VLDB Endow. **14**(8), 1441–1453 (2021)
27. Zhang, F., Guo, H., Ouyang, D., Yang, S., Lin, X., Tian, Z.: Size-constrained community search on large networks: An effective and efficient solution. IEEE Transactions on Knowledge and Data Engineering (2023)
28. Zhu, Z.A., Lattanzi, S., Mirrokni, V.S.: A local algorithm for finding well-connected clusters. In: ICML. vol. 28, pp. 396–404 (2013)