Scalable and Effective Conductance-Based Graph Clustering

Longlong Lin[1], Rong-Hua Li [2], Tao Jia[1]

[1]College of Computer and Information Science, Southwest University, Chongqing 400715, China
[2]Beijing Institute of Technology, China
longlonglin@swu.edu.cn; lironghuabit@126.com; tjia@swu.edu.cn

Abstract

Conductance-based graph clustering has been recognized as a fundamental operator in numerous graph analysis applications. Despite the significant success of conductance-based graph clustering, existing algorithms are either hard to obtain satisfactory clustering qualities, or have high time and space complexity to achieve provable clustering qualities. To overcome these limitations, we devise a powerful *peeling*-based graph clustering framework *PCon*. We show that many existing solutions can be reduced to our framework. Namely, they first define a score function for each vertex, then iteratively remove the vertex with the smallest score. Finally, they output the result with the smallest conductance during the peeling process. Based on our framework, we propose two novel algorithms *PCon_core* and *PCon_de* with linear time and space complexity, which can efficiently and effectively identify clusters from massive graphs with more than a few billion edges. Surprisingly, we prove that *PCon_de* can identify clusters with near-constant approximation ratio, resulting in an important theoretical improvement over the well-known quadratic Cheeger bound. Empirical results on real-life and synthetic datasets show that our algorithms can achieve 5∼42 times speedup with a high clustering accuracy, while using 1.4∼7.8 times less memory than the baseline algorithms.

Introduction

Graph clustering is an important algorithmic primitive with applications in numerous tasks, including image segmentation (Shi and Malik 1997; Tolliver and Miller 2006), community detection (Fortunato 2009; Leskovec, Lang, and Mahoney 2010), and machine learning (Belkin and Niyogi 2001; Bianchi, Grattarola, and Alippi 2020). Generally, graph clustering aims to partition the entire graph into several non-overlapping vertex sets, called clusters, such that the vertices within the same cluster have more connections than the vertices in different clusters. Many graph clustering methods have been proposed, such as modularity based graph clustering (Newman 2004), structural graph clustering (Xu et al. 2007), and cohesive subgraph based clustering (Chang and Qin 2019). Perhaps, the most representative graph clustering method is the *conductance*-based clustering (Andersen, Chung, and Lang 2006; Yang et al. 2019) due to its nice structure properties and solid theoretical foundation.

Given an undirected graph $G(V, E)$, the *conductance* of the cluster C is defined as $\phi(C) = \frac{|E(C,\bar{C})|}{\min\{vol(C),2m-vol(C)\}}$, in which $|E(C,\bar{C})|$ is the number of edges with one endpoint in C and another not in C, $vol(C)$ is the sum of the degree of all vertices in C, and m is the number of the edges in G. Thus, a smaller $\phi(C)$ implies that the number of edges going out of C is relatively small compared with the number of edges within C. As a consequence, the smaller the $\phi(C)$, the better the clustering quality of the cluster C (Leskovec, Lang, and Mahoney 2010; Yang and Leskovec 2015). Therefore, *conductance*-based graph clustering aims at identifying a vertex set C with the smallest $\phi(C)$. However, identifying the smallest conductance $\phi^* \in (0, 1]$ raises significant challenges due to its NP-hardness (Chawla et al. 2006). On the positive side, many approximate or heuristic algorithms have been proposed to either reduce the conductance of the returned cluster or improve the efficiency. For example, classic *Fiedler* vector-based spectral clustering algorithm outputs a cluster with conductance $O(\sqrt{\phi^*})$ (Alon and Milman 1985). Unfortunately, such an algorithm has to compute the eigenvector corresponding to the second smallest eigenvalue of normalized Laplacian matrix of G, resulting in prohibitively high time and space complexity. To boost the efficiency, numerous diffusion-based local clustering algorithms have been proposed, whose running time depends only on the size of the resulting cluster and is independent of or depends at most polylogarithmically on the size of the entire graph (Spielman and Teng 2004; Andersen, Chung, and Lang 2006; Kloster and Gleich 2014). However all these diffusion-based local clustering algorithms are heuristic and the clustering quality of their output is heavily dependent on many hard-to-tune parameters, resulting in that their clustering performance are unstable and in most cases very poor (Zhu, Lattanzi, and Mirrokni 2013; Fountoulakis, Wang, and Yang 2020). Thus, it is very challenging to improve both the computational efficiency and the clustering quality.

To this end, we propose a powerful *peeling*-based computing framework *PCon*, which can efficiently and effectively identify conductance-based clusters. In particular, we observe that *Fiedler* vector-based spectral clustering algorithms and diffusion-based local clustering algorithms are essentially a *peeling*-based computing paradigm. Namely, they first define a score function for each vertex, then iter-

atively remove the vertex with the smallest score. Finally, they output the results with the smallest conductance during the peeling process. Thus, the primary challenge is how to design a proper score function that is easier to compute and can obtain better clustering quality. With the help of the concept of degeneracy ordering (Batagelj and Zaversnik 2003), we propose a heuristic algorithm *PCon_core* with linear time and space complexity. Specifically, *PCon_core* assigns the score of each vertex as the vertex's degeneracy ordering. To further improve the clustering quality, we devise another effective algorithm *PCon_de*, which assigns the score of each vertex based on the *degree ratio* (Definition 3). *Degree ratio* is a novel concept, which is the ratio of the degree of the vertex in the current subgraph to the degree of the vertex in the original graph. Surprisingly, we prove that *PCon_de* has a near-constant approximation ratio, which achieves an important theoretical improvement over the well-known quadratic chegger bound of *Fiedler* vector-based spectral clustering. In a nutshell, our main contributions are listed as follows.

- **Novel Computing Framework:** We introduce *PCon*, a novel *peeling*-based graph clustering framework, which can embrace most state-of-the-art algorithms.

- **Novel algorithms:** Based on *PCon*, we develop two novel and efficient algorithms with linear time and space complexity. One is a heuristic algorithm which aims to optimize efficiency, while the other is an approximation algorithm which can obtain provable clustering qualities.

- **Extensive Experiments:** We conduct extensive experiments on eleven datasets with six competitors to test the scalability and effectiveness of the proposed algorithms. The experimental results show that our proposal outperforms diffusion-based local clustering by a large margin in terms of clustering accuracy, and achieves 5∼42 times speedup with a high clustering accuracy while using 1.4∼7.8 times less memory than *Fiedler* vector-based spectral clustering.

Problem Formulation

We use $G(V, E)$ to denote an undirected graph, in which V (resp. E) indicates the vertex set (resp. the edge set) of G. Let $|V| = n$ and $|E| = m$ be the number of vertices and the number of edges, respectively. For simplicity, in this paper, we focus on un-weighted graphs[1]. Let $G_S = (S, E_S)$ be the subgraph induced by S if $S \subseteq V$ and $E_S = \{(u, v) \in E | u, v \in S\}$. We use $N_S(v) = \{u \in S | (u, v) \in E\}$ to denote the neighbors of v in S. Let $d_S(v) = |N_S(v)|$ be the degree of vertex v in S. When the context is clear, we use $N(v)$ and $d(v)$ to represent $N_V(v)$ and $d_V(v)$, respectively. For a vertex subset $S \subseteq V$, we define $\bar{S} = V \setminus S$ as the complement of S. We use $E(S, \bar{S}) = \{(u, v) \in E | u \in S, v \in \bar{S}\}$ to represent the edges with one endpoint in S and another not in S. Let $vol(S) = \sum_{u \in S} d(u)$ be the sum of the degree of all vertices in S.

A cluster is a vertex set $S \subseteq V$. According to (Leskovec, Lang, and Mahoney 2010; Yang and Leskovec 2015), we

---

[1]Because our results can be easily extended to the weighted case in a straightforward manner.

Clusters	Conductance
{v₁}	2/min{2,20-2}=1
{v₁,v₂}	3/min{5,20-5}=3/5
{v₁,v₂,v₃}	3/min{9,20-9}=3/11
{v₁,v₂,v₃,v₄}	1/min{11,20-11}=1/9
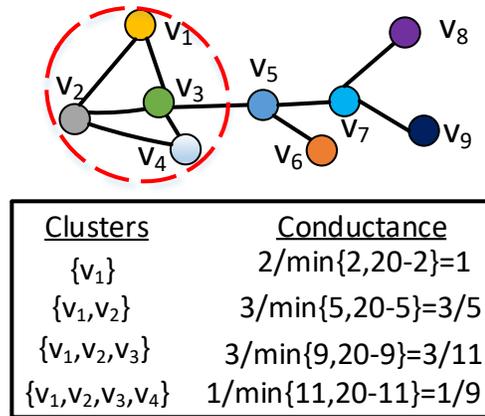
Figure 1: Illustration of conductance for a synthetic graph with 9 vertices and 10 edges. The conductance of several clusters is shown in the box. The smallest conductance $\phi^* = 1/9$ and the corresponding cluster is $\{v_1, v_2, v_3, v_4\}$ included in the red circle.

know that a cluster S is good if the cluster is densely connected internally and well separated from the remainder of G. Therefore, we use a representative metric *conductance* (Fiedler 1973; Spielman and Teng 2004; Andersen, Chung, and Lang 2006; Kloster and Gleich 2014) to measure the quality of a cluster S.

Definition 1 *Given an undirected graph $G(V, E)$ and a vertex subset $S \subsetneq V$, the conductance of S is defined as* $\phi(S) = \frac{|E(S, \bar{S})|}{\min\{vol(S), 2m - vol(S)\}}$. $\phi(V) = 1$ *for convenience.*

By Definition 1, we have $\phi(S) = \phi(\bar{S})$. Figure 1 illustrates the concept of *conductance* on a synthetic graph.

Problem Statement. Given an undirected graph $G(V, E)$, the goal of the *conductance*-base graph clustering is to identify a vertex subset $S^* \subseteq V$, satisfying $vol(S^*) \leq m$ and $\phi(S^*) \leq \phi(S)$ for any $S \subseteq V$. For simplicity, we use ϕ^* to represent $\phi(S^*)$.

Existing Solutions

Here, we review some state-of-the-art algorithms. For convenience, we classify these algorithms into three categories: *Fiedler* vector-based spectral clustering, diffusion-based local clustering, and other methods.

Fiedler Vector-Based Spectral Clustering

Fiedler vector-based spectral clustering obtains the clustering result by calculating the spectrum of the normalized Laplacian matrix. Specifically, let A be the adjacency matrix of G, where $A_{uv} = 1$ if $(u, v) \in E$, and $A_{uv} = 0$ otherwise. We use D to represent the diagonal degree matrix of G, in which $D_{uu} = d(u)$. The Laplacian matrix of G is defined as $L = D - A$. Furthermore, the normalized Laplacian matrix is defined as $\mathcal{L} = D^{-1/2} L D^{-1/2}$. The following theorem is an important theoretical basis of *Fiedler* vector-based spectral clustering.

Methods	Accuracy Guarantee	Time Complexity	Remark
SC (Fiedler 1973)	$O(\sqrt{\phi^*})$	$O(n^3)$	Eigenvector-based
ASC (Trevisan 2017)	$O(\sqrt{4\phi^* + 2\epsilon})$	$O((m+n)\frac{1}{\epsilon}\log\frac{n}{\epsilon})$	Eigenvector-based
NIBBLE (Spielman and Teng 2004)	$\times$	$O(\frac{\log\frac{1}{\epsilon}}{\epsilon})$	Diffusion-based
NIBBLE_PPR (Andersen, Chung, and Lang 2006)	$\times$	$O(\frac{\log\frac{1}{\alpha\epsilon}}{\alpha\epsilon})$	Diffusion-based
HK_Relax (Kloster and Gleich 2014)	$\times$	$O(\frac{te^t\log(\frac{1}{\epsilon})}{\epsilon})$	Diffusion-based
PCon_core (this paper)	$\times$	$O(m+n)$	Degeneracy-based
PCon_de (this paper)	$O(1/2 + 1/2\phi^*)$	$O(m+n)$	Degree Ratio-based

Table 1: A comparison of state-of-the-art algorithms. ϕ^* is the smallest conductance value and $\phi^* \in (0, 1]$. ϵ is the error tolerance. α and t are model parameters of *NIBBLE_PPR* and *HK_Relax*, respectively. The space complexity of SC is $O(n^2)$ and the rest is $O(m+n)$. $\times$ represents the corresponding method has no accuracy guarantee.

Theorem 1 (Cheeger inequality (Alon and Milman 1985)) *Given an undirected graph $G(V, E)$ and its normalized Laplacian matrix $\mathcal{L}$, we assume that λ_2 is the second smallest eigenvalue of $\mathcal{L}$. Then, we have $\lambda_2/2 \leq \phi^* \leq \sqrt{2\lambda_2}$.*

At a high level, *Fiedler* vector-based spectral clustering consists of three steps: (1) Compute the eigenvector x of λ_2. (2) Sort all entries in x such that $x_1 \leq x_2 \leq ... \leq x_n$. (3) Output $S = \arg\min \phi(S_i)$, in which $S_i = \{x_1, x_2, ..., x_i\}$.

Theorem 2 ((Fiedler 1973; Alon and Milman 1985)) *Let S be the vertex subset returned by Fiedler vector-based spectral clustering, we have $\phi(S) \leq \sqrt{2\lambda_2}$. Furthermore, we have $\phi^* \leq \phi(S) \leq 2\sqrt{\phi^*}$.*

Note that since *Fiedler* vector-based spectral clustering (*SC* for short) needs to calculate the eigenvector of $\mathcal{L}$, its time complexity is $O(n^3)$ and space complexity is $O(n^2)$ (Fiedler 1973), resulting in poor scalability. Thus, some authors devised efficient approximate algorithms. For example, Trevisan proposed *ASC* to approximate the eigenvector of second smallest eigenvalue of $\mathcal{L}$ (Trevisan 2017). As a result, *ASC* identifies a cluster S with $\phi(S) \leq \sqrt{4\phi^* + 2\epsilon}$ using $O((m+n) \cdot \frac{1}{\epsilon}\log\frac{n}{\epsilon})$ time cost, in which ϵ is the error tolerance. Although the time complexity of *ASC* is much less than of *SC*, the quality of *ASC* decreases rapidly as shown in our experiments.

Diffusion-based Local Clustering

Diffusion-based local clustering is another technique that propagates information from the given query seed vertex q to identify clusters. Here, we state three state-of-the-art graph diffusions: Truncated Random Walk (Spielman and Teng 2004), Personalized PageRank (Andersen, Chung, and Lang 2006), and Heat Kernel (Kloster and Gleich 2014). For simplicity, we use π_T, π_P, and π_H to represent the probability distribution at the end of Truncated Random Walk, Personalized PageRank, and Heat Kernel, respectively.

Before proceeding further, we give some important notations. Let $P = D^{-1}A$ be the probability transition matrix of G, in which $P_{uv} = 1/d(u)$ for any $v \in N(u)$. Moreover, we use P^k to represent the k-hop probability transition matrix of G. Namely, P^k_{uv} is the probability that a k-hop ($k \geq 1$) random walk from vertex u would end at vertex v.

1) Truncated Random Walk (Spielman and Teng 2004) is a graph diffusion algorithm where propagation and truncation are alternately performed. Specifically, for any vector s and error tolerance ϵ, we define a truncation operator $Tr(s)$ on u as follows:

$$Tr(s)[u] = \begin{cases} s[u], & if \quad s[u] \geq d(u)\epsilon \\ 0, & Otherwise \end{cases} \quad (1)$$

Furthermore, we use $Z_0 = \chi_q$ to represent the one-hot vector with only a value-1 entry corresponding to the given query seed q. The propagation and truncation are denoted as $Z_i = Tr(Z_{i-1}P)$, in which $Z_{i-1}P$ is the propagation process and $Tr(Z_{i-1}P)$ is the truncation process which can obtain the next probability distribution. Thus, we let $\pi_T = Z_N$ be the probability distribution after N iterations, in which N is an input parameter.

2) Personalized PageRank (Andersen, Chung, and Lang 2006) models a special random walk process. Specifically, given a stop probability parameter α (a.k.a. teleportation probability), we denote a α-discount random walk as follows: (1) It starts from the given query seed q. (2) At each step it stops in the current vertex with probability α, or it continues to walk according to the probability transition matrix P with 1-α. Thus, $\pi_P(u)$ is the probability that the α-discount random walk stops in u.

3) Heat Kernel (Kloster and Gleich 2014) also models a special random walk process. Specifically, given a heat constant t, $\pi_H(u)$ is the probability that a random walk of length k starting from the given query seed q would end at the vertex u, in which k is sampled from the Poisson distribution $\eta(k) = \frac{e^{-t}t^k}{k!}$. Thus $\pi_H(u) = \sum_{k=0}^{\infty} \eta(k)P^k_{qu}$.

Similarly, these diffusion-based local clustering algorithms also consist of three steps: (1) Compute the probability distribution π at the end of the corresponding graph diffusion, and $y = \pi D^{-1}$. (2) Sort all non-zero entries in y such that $y_1 \geq y_2 \geq ... \geq y_{sup(y)}$, in which $sup(y)$ is the number of the non-zero entries in y. (3) Output $S = \arg\min \phi(S_i)$, in which $S_i = \{y_1, y_2, ...y_i\}$.

Note that since diffusion-based local clustering algorithms aim to recover the cluster to which the given query

seed q belongs, they only have locally-biased Cheeger-like quality (Spielman and Teng 2004; Andersen, Chung, and Lang 2006; Kloster and Gleich 2014). Namely, diffusion-based local clustering algorithms do not give the theoretical gap to ϕ^*. Besides, the clustering quality of their output is heavily dependent on the given query seed and hard-to-tune parameters, resulting in that they are unstable and prone to finding degenerate solutions in most cases (Zhu, Lattanzi, and Mirrokni 2013; Fountoulakis, Wang, and Yang 2020).

Other Methods

Optimization-based (Leighton and Rao 1999; Arora, Rao, and Vazirani 2004) and flow-based (Orecchia and Zhu 2014; Veldt, Gleich, and Mahoney 2016; Wang et al. 2017) are two important techniques for improving *Fiedler* vector-based spectral clustering and diffusion-based local clustering. For example, Leighton et al. (Leighton and Rao 1999) used linear programming to solve the conductance-based graph clustering with $O(\log n)$-approximation. Furthermore, Arora et al. (Arora, Rao, and Vazirani 2004) achieved the best $O(\sqrt{\log n})$-approximation using a non-trivially semi-definite programming algorithm with $O(\frac{(n+m)^2}{\epsilon^{O(1)}})$ time cost. However, all these methods are mostly of theoretical interests only, as they are difficult to implement and offer rather poor practical efficiency (Yang et al. 2019).

The Proposed Algorithms

Here, we first present a three-stage computing framework, which embraces the computing paradigm of most existing approaches. Then, we develop two scalable and effective algorithms to solve the *conductance*-based graph clustering.

Computing Framework

Inspired by most existing approaches (Fiedler 1973; Spielman and Teng 2004; Andersen, Chung, and Lang 2006; Kloster and Gleich 2014), we propose a three-stage computing framework *PCon*. In *Stage 1*, we give a pre-defined score function for every vertex according to the corresponding applications. For simplicity, we use $s(u)$ to represent the score of vertex u. In *Stage 2*, we iteratively remove the vertex with the smallest score. Such an iterative deletion process is referred to as a peeling process. In *Stage 3*, we output the result with smallest conductance during the *peeling* process.

Obviously, *Stage 1* is key to our proposed computing framework. Different algorithms have different score functions. For example, $s(u) = x[u]$ for *Fiedler* vector-based spectral clustering, in which x is the eigenvector of second smallest eigenvalue of $\mathcal{L}$. In diffusion-based local clustering, we can know that $s(u) = y[u]$, in which $y = \pi D^{-1}$ and π is the probability distribution of the diffusion process. Thus, most state-of-the-art algorithms can be reduced to the proposed three-stage computing framework. However, the primary problem with these existing algorithms to partition the graph is that it is difficult to efficiently and effectively obtain the score function. To this end, in the next sections, we devise two new score functions, which are easier to compute.

Algorithm 1: *PCon_core*

Input: An undirected graph $G(V, E)$
Output: A cluster S

1: $\{u_1, u_2, ..., u_n\} \leftarrow$ the degeneracy ordering
2: $i \leftarrow n, S_i \leftarrow V, S \leftarrow V$
3: **while** $i \neq 0$ **do**
4:    **if** $vol(S_i) \leq m$ and $\phi(S_i) < \phi(S)$ **then**
5:       $S \leftarrow S_i$
6:    **end if**
7:    $S_{i-1} \leftarrow S_i \setminus \{u_i\}, i \leftarrow i - 1$
8: **end while**
9: **return** S
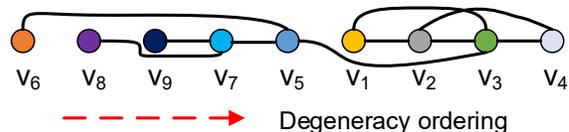
Figure 2: A degeneracy ordering of vertices in Figure 1.

The *PCon_core* Algorithm

Recall that in *Stage 2*, we need to iteratively remove the vertex with the smallest score. Namely, we have to create a linear ordering on vertices. Many ordering strategies have been proposed for numerous graph analysis tasks, such as graph coloring (Hasenplaugh et al. 2014) and k-clique listing (Li et al. 2020). However, ordering techniques for conductance-based graph clustering are less explored. To fill this gap, we propose a *simple* algorithm with the help of the well-known degeneracy ordering (Batagelj and Zaversnik 2003).

Definition 2 (Degeneracy ordering) *Given an undirected graph G, a permutation $(u_1, u_2, ..., u_n)$ on all vertices of G is a degeneracy ordering iff every vertex u_i has the minimum degree in the subgraph of G induced by $\{u_i, u_{i+1}, ..., u_n\}$, that is, $u_i = \arg\min\{d_{V_i}(u) | u \in V_i\}$ where $V_i = \{u_i, u_{i+1}, ..., u_n\}$.*

We use degeneracy ordering to assign the score $s(u)$ of vertex u. Specifically, u is the $s(u)$-th element in the degeneracy ordering from left to right.

Example 1 *Reconsider the graph in Figure 1. According to Definition 2, we can derive the degeneracy ordering as $\{v_6, v_8, v_9, v_7, v_5, v_1, v_2, v_3, v_4\}$ which is illustrated in Figure 2. Consider a vertex v_1, we can know that $s(v_1) = 6$. This is because that v_1 has the minimum degree in the subgraph of G induced by $\{v_1, v_2, v_3, v_4\}$. Similarity, we have $s(v_6) = 1$ and $s(v_7) = 4$.*

The degeneracy ordering can be efficiently computed within linear time by the classic core-decomposition (Batagelj and Zaversnik 2003). Specifically, it iteratively removes the vertex with the smallest degree in the current subgraph until all vertices are removed. When a vertex is removed, the degree of other vertices is updated accordingly. As a consequence, the sequence of the removed vertices forms the degeneracy ordering. Using a proper data structure (e.g., bin-sort), we can implement the above vertex-removal

process in linear time (Batagelj and Zaversnik 2003). Based on the degeneracy ordering, we propose a *simple but practically effective* algorithm *PCon_core*, which is outlined in Algorithm 1. Specifically, we first set the score $s(u)$ for each vertex $u \in V$ according to the degeneracy ordering (Line 1). Then, we execute *Stage 2* (Lines 3, 7, and 8) and *Stage 3* (Lines 4-6 and 9).

The *PCon_de* Algorithm

As stated in the Introduction, *Fiedler* vector-based spectral clustering can obtain a cluster with conductance of $O(\sqrt{\phi^*})$. However, *Fiedler* vector-based spectral clustering has prohibitively high time and space complexity. On the other hand, although diffusion-based local clustering has a very low time complexity, it is heuristic without a global conductance guarantee. Thus, an interesting problem is to devise an algorithm that has better time complexity than *Fiedler* vector-based spectral clustering and better conductance quality than diffusion-based local clustering. To this end, we propose a novel algorithm *PCon_de* with linear time and space complexity, which has a near-constant approximation ratio.

Lemma 1 *Given an undirected graph $G(V, E)$ and a vertex subset S, we have $\phi(S) = 1 - \frac{\sum_{u \in S} d_S(u)}{\sum_{u \in S} d_V(u)}$ if $vol(S) \leq m$.*

Proof 1 *According to Definition 1, if $vol(S) \leq m$, we have $\phi(S) = \frac{|E(S,\bar{S})|}{\min\{vol(S), 2m - vol(S)\}} = \frac{|E(S,\bar{S})|}{vol(S)}$. Furthermore, $\frac{|E(S,\bar{S})|}{vol(S)} = \frac{\sum_{u \in S} d_{\bar{S}}(u)}{\sum_{u \in S} d_V(u)} = \frac{\sum_{u \in S} d_{\bar{S}}(u) + \sum_{u \in S} d_V(u) - \sum_{u \in S} d_V(u)}{\sum_{u \in S} d_V(u)} = \frac{\sum_{u \in S} d_V(u) - \sum_{u \in S} d_S(u)}{\sum_{u \in S} d_V(u)}$. Thus, $\phi(S) = 1 - \frac{\sum_{u \in S} d_S(u)}{\sum_{u \in S} d_V(u)}$.*

For simplicity, we denote a function $g(S) = \frac{\sum_{u \in S} d_S(u)}{2 \sum_{u \in S} d_V(u)}$ and assume that the larger the value of $g(S)$, the better the quality of S. Generally, we let $\widetilde{S}$ be the optimal vertex set for $g(.)$. That is, $g(\widetilde{S}) \geq g(S)$ for any vertex subset $S \subseteq V$.

Lemma 2 *Given an undirected graph $G(V, E)$, we have $\frac{d_{\widetilde{S}}(u)}{d_V(u)} \geq g(\widetilde{S})$ for any $u \in \widetilde{S}$ holds.*

Proof 2 *This lemma can be proved by contradiction. Assume that there is a vertex $u \in \widetilde{S}$ such that $\frac{d_{\widetilde{S}}(u)}{d_V(u)} < g(\widetilde{S})$, we have $d_{\widetilde{S}}(u) < g(\widetilde{S})d_V(u)$. Thus, $g(\widetilde{S} \setminus \{u\}) = \frac{1/2 \sum_{v \in \widetilde{S}} d_{\widetilde{S}}(v) - d_{\widetilde{S}}(u)}{\sum_{v \in \widetilde{S}} d_V(v) - d_V(u)} > \frac{1/2 \sum_{v \in \widetilde{S}} d_{\widetilde{S}}(v) - g(\widetilde{S})d_V(u)}{\sum_{v \in \widetilde{S}} d_V(v) - d_V(u)} = \frac{g(\widetilde{S}) \sum_{v \in \widetilde{S}} d_V(v) - g(\widetilde{S})d_V(u)}{\sum_{v \in \widetilde{S}} d_V(v) - d_V(u)} = g(\widetilde{S})$, which contradicts that $\widetilde{S}$ is the optimal vertex set for $g(.)$. As a result, $\frac{d_{\widetilde{S}}(u)}{d_V(u)} \geq g(\widetilde{S})$ for any $u \in \widetilde{S}$ holds.*

Definition 3 (Degree ratio) *Given an undirected graph $G(V, E)$ and a subgraph G_S, the degree ratio of $u \in S$ w.r.t. G and G_S is defined as $Dr_S(u) = \frac{d_S(u)}{d_V(u)}$.*

Algorithm 2: *PCon_de*

Input: An undirected graph $G(V, E)$
Output: A cluster $\hat{S}$

1: $S \leftarrow V; \hat{S} \leftarrow V$
2: $Dr_S(u) \leftarrow \frac{d_S(u)}{d_V(u)}$ for each vertex $u \in S$
3: **while** $S \neq \emptyset$ **do**
4:    $u \leftarrow \arg \min\{Dr_S(u) | u \in S\}$
5:    $S \leftarrow S \setminus \{u\}$
6:    **if** $g(S) > g(\hat{S})$ and $vol(S) \leq m$ **then**
7:       $\hat{S} \leftarrow S$
8:    **end if**
9:    **for** $v \in N_S(u)$ **do**
10:       $Dr_S(v) \leftarrow Dr_S(v) - \frac{1}{d_V(v)}$
11:    **end for**
12: **end while**
13: **return** $\hat{S}$

Based on the above lemmas and definitions, we devise a linear time greedy removing algorithm called *PCon_de*, which is shown in Algorithm 2. In particular, Algorithm 2 first initializes the current search space S as V, candidate result $\hat{S}$ as V, and the *degree ratio* $Dr_S(u)$ for every vertex $u \in S$ according to Definition 3 (Lines 1-2). Subsequently, it executes the greedy removing process in each round to improve the quality of the target cluster (Lines 3-12). Specifically, in each round, it obtains one vertex u with the smallest *degree ratio* (Line 4). Lines 5-11 update the current search space S, the candidate result $\hat{S}$ if any, and $Dr_S(v)$ for $v \in N_S(u)$. The iteration terminates once the current search space is empty (Line 3). Finally, it returns $\hat{S}$ as the approximate solution (Line 13).

Theorem 3 *Algorithm 2 can identify a cluster with conductance $1/2 + 1/2\phi^*$.*

Proof 3 *Let $\widetilde{S}$ is the optimal vertex set for $g(.)$. In Lines 3-12, Algorithm 2 executes the peeling process. That is, in each round, it greedily deletes the vertex with the smallest degree ratio. Consider the round t when the first vertex u of $\widetilde{S}$ is deleted. Let V_t be the vertex set from the beginning of round t. Clearly, $\widetilde{S}$ is the subset of V_t because u is the first deleted vertex of $\widetilde{S}$. This implies that $\min_{v \in V_t} Dr_{V_t}(v) = Dr_{V_t}(u) = \frac{d_{V_t}(u)}{d_V(u)} \geq \frac{d_{\widetilde{S}}(u)}{d_V(u)}$. Furthermore, $g(V_t) = \frac{\sum_{v \in V_t} d_{V_t}(v)}{2 \sum_{v \in V_t} d_V(v)} = \frac{\sum_{v \in V_t} d_V(v) \frac{d_{V_t}(v)}{d_V(v)}}{2 \sum_{v \in V_t} d_V(v)} \geq \frac{\sum_{v \in V_t} d_V(v) \frac{d_{V_t}(u)}{d_V(u)}}{2 \sum_{v \in V_t} d_V(v)} = \frac{d_{V_t}(u)}{2 d_V(u)} \geq \frac{d_{\widetilde{S}}(u)}{2 d_V(u)}$. By Lemma 2, we have $\frac{d_{\widetilde{S}}(u)}{d_V(u)} \geq g(\widetilde{S})$. Thus, $g(V_t) \geq \frac{g(\widetilde{S})}{2}$. Since Algorithm 2 maintains the optimal solution during the peeling process in Lines 6-8, $\hat{S}$ will be returned in Line 13 and $g(\hat{S}) \geq g(V_t) \geq \frac{g(\widetilde{S})}{2}$. On the other hand, by Lemma 1, we have $g(\hat{S}) = \frac{1 - \phi(\hat{S})}{2}$ and $g(S^*) = \frac{1 - \phi(S^*)}{2}$. According to the definition of $\widetilde{S}$, we know that $g(\widetilde{S}) \geq g(S^*)$. Thus,*

Dataset	$|V|$	$|E|$	$\bar{d}$
DBLP	317,080	1,049,866	6.62
Youtube	1,134,890	2,987,624	5.27
Pokec	1,632,803	22,301,964	2.73
LJ	4,843,953	42,845,684	17.69
Orkut	3,072,441	117,185,083	76.28
Twitter	41,652,231	1,202,513,046	57.74

Table 2: Dataset statistics. $\bar{d}$ is the average degree.

$\frac{1-\phi(\hat{S})}{2} = g(\hat{S}) \geq \frac{g(\widetilde{S})}{2} \geq \frac{g(S^*)}{2} = \frac{1-\phi(S^*)}{4}$. *Namely,* $\phi(\hat{S}) \leq 1 - \frac{1-\phi(S^*)}{2} = 1/2 + 1/2\phi^*$. *As a result, Algorithm 2 can identify a cluster with conductance* $1/2 + 1/2\phi^*$.

Empirical Results

Experimental Setup

We evaluate our proposed solutions on six real-life publicly-available datasets[2] (Table 2), which are widely used benchmarks for conductance-based graph clustering (Shun et al. 2016; Yang et al. 2019). The maximum connected components of these datasets are used in the experiments. We also use five synthetic graphs *LFR* (Lancichinetti, Fortunato, and Kertész 2009), *WS* (Watts and Strogatz 1998), *PLC* (Holme and Kim 2002), *ER* (Erdos, Rényi et al. 1960), and *BA* (Barabási and Albert 1999). The following six competitors are implemented for comparison.

- Eigenvector-based methods: *SC* (Fiedler 1973) and *ASC* (Trevisan 2017). *SC* (resp. *ASC*) used the exact eigenvector (resp. approximate eigenvector) of the second smallest eigenvalue of $\mathcal{L}$ to execute the *peeling* process. We use sparse matrices to highly optimize these algorithms.

- Diffusion-based methods: *NIBBLE_PPR* (Andersen, Chung, and Lang 2006) and *HK_Relax* (Kloster and Gleich 2014). Since *NIBBLE_PPR* and *HK_Relax* took a seed vertex as input, to be more reliable, we randomly select 50 vertices as seed vertices and report the average runtime and quality. Unless specified otherwise, following previous work (Shun et al. 2016), we set $\alpha = 0.01$ and $\epsilon = \frac{1}{m}$ for *NIBBLE_PPR*; $t = 10$ and $\epsilon = \frac{1}{m}$ for *HK_Relax*. Note that we do not include *NIBBLE* of Table 1 in the experiments because *NIBBLE* is outperformed by *NIBBLE_PPR* and *HK_Relax* (Shun et al. 2016).

- Flow-based methods: *SimpleLocal* (Veldt, Gleich, and Mahoney 2016) and *CRD* (Wang et al. 2017). These methods devise specialized max-flow algorithms with early termination. We take their default parameters in our experiments.

Results on Real-World Graphs

Table 3 reports runtime and conductance for each method on real-world graphs. We have the following observations: (1) The proposed algorithms *PCon_core* and *PCon_de* are

---

[2]All datasets can be downloaded from http://snap.stanford.edu/
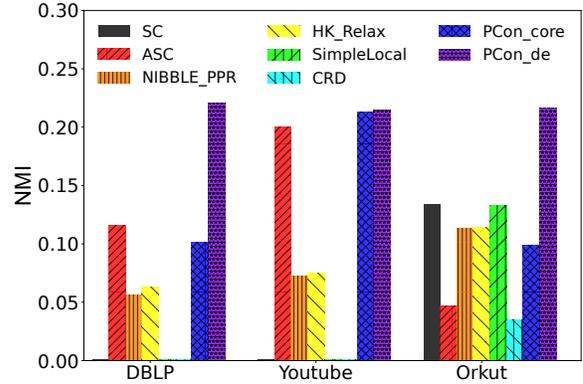
Figure 3: *NMI* scores on real-world graphs with ground-truth clusters.
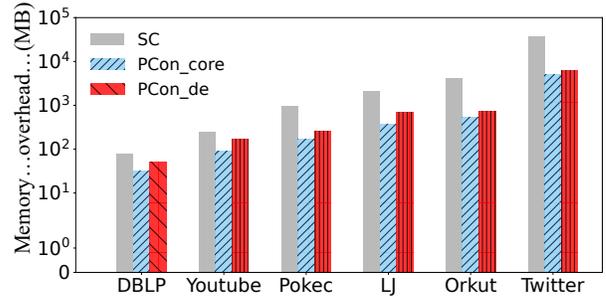
Figure 4: Memory overhead on real-world graphs (excluding the size of the graph itself).

consistently faster than other methods on most graphs, especially for larger graphs. In particular, they achieve the speedups of 5, 14, 42, and 17 times over *SC* on DBLP, Youtube, LJ, and Twitter, respectively. For example, on Twitter with more than a few billion edges, *PCon_core* and *PCon_de* take 5,486 seconds and 7,655 seconds to obtain the result, respectively, while *SC* takes 130,570 seconds. (2) *PCon_core* is slightly fast than *PCon_de* but *PCon_core* has poor conductance value. (3) The clustering quality returned by *PCon_de* outperforms other methods on five of the six datasets. Besides, *PCon_de* always outperforms *NIBBLE_PPR* and *HK_Relax*. This is because *PCon_de* can find clusters with near-liner approximation ratio, while *Fiedler* vector-based spectral clustering algorithms (i.e., *SC* and *ASC*) have quadratic chegger bound and diffusion-based local clustering algorithms (i.e., *NIBBLE_PPR* and *HK_Relax*) have no guarantee of clustering quality (Table 1). Therefore, these results give clear evidences that the proposed algorithms can achieve significant speedup with high clustering quality compared with the baselines.

We use the normalized mutual information (*NMI* for short) (Cilibrasi and Vitányi 2005) to measure how "close" each detected cluster is to the ground-truth one. Note that for a cluster C, the larger the *NMI*, the better the quality of the cluster C. Figure 3 shows *NMI* scores of different methods on real-world graphs with ground-truth clusters. As can be seen, *PCon_de* consistently outperforms other methods. The *NMI* scores of all methods other than *PCon_de* vary signif-

Runtime/Conductance	DBLP	Youtube	Pokec	LJ	Orkut	Twitter
SC	56/**0.009**	552/0.006	165/0.002	14492/0.001	837/0.007	130570/ 0.002
ASC	21/0.482	73/0.554	730/0.454	1185/0.423	3089/0.415	67409/0.602
NIBBLE_PPR	**7**/0.130	**18**/0.110	156/0.184	784/0.021	3768/0.009	5822/0.177
HK_Relax	31/0.127	138/0.113	1271/0.011	2144/0.036	6016/0.008	99613/0.026
SimpleLocal	108/**0.009**	596/0.006	213/0.002	14807/0.001	16267/0.013	283745/0.004
CRD	47/0.184	235/0.183	1231/0.218	1279/0.194	7261/0.141	38060/0.336
PCon_core	9/0.106	27/0.404	**116**/0.327	**266**/0.088	**500**/0.223	**5486**/0.419
PCon_de	11/0.027	39/**0.004**	141/**0.001**	345/**0.000**	577/**0.006**	7655/**0.000**

Table 3: Runtime (in seconds) and conductance on real-world graphs. The best result in each metric is highlighted in bold.

(a) Scalability testing on *ER* synthetic graph (b) *LFR* synthetic graphs varying μ (c) *LFR* synthetic graphs varying n
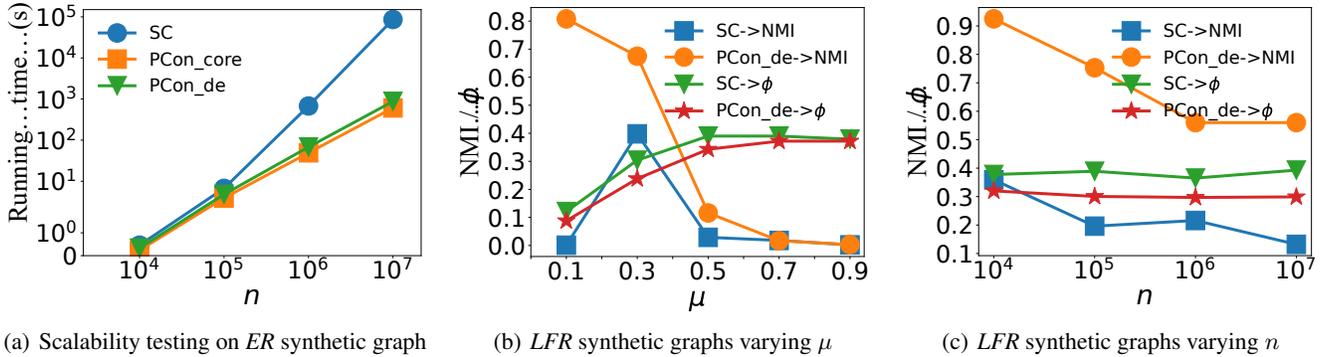
Figure 5: Results on synthetic graphs.

icantly depending on the dataset. For example, *SC*, *SimpleLocal*, and *CRD* are almost zero on DBLP and Youtube, but they have relatively good *NMI* scores on Orkut. Therefore, these results imply that *PCon_de* approximates the ground-truth clusters more effectively than all other methods.

From Figure 4, we can see that memory overhead of *PCon_core* and *PCon_de* are consistently less than *SC* on all datasets. In particular, they save $1.4 \sim 7.8$ times memory overhead compared to *SC*. For example, on Orkut graph, *PCon_core* and *PCon_de* consume about 535.88MB and 746.26MB to obtain the result, respectively, while *SC* consumes 4209.25MB. This is because *PCon_core* and *PCon_de* only need linear space complexity to calculate the score of each vertex. However, *SC* adopts the eigenvector of the matrix to compute the score of each vertex, thus it requires squared space complexity in the worst case. These results demonstrate that *PCon_core* and *PCon_de* can identify clusters with low memory.

Results on Synthetic Graphs

We further use synthetic graphs to evaluate the scalability and effectiveness of our proposed solutions. In particular, we only present the scalability for *ER* (Erdos, Rényi et al. 1960) with varying the number of vertices in Figure 5(a), but all other synthetic graphs have similar trends. As can be seen, *PCon_core* and *PCon_de* scale near-linear with respect to the size of the graphs. However, *SC* has poor scalability because its time cost fluctuates greatly as the graph size increases. These results indicate that our proposed algorithms can handle massive graphs while *SC* cannot.

The *LFR* model (Lancichinetti, Fortunato, and Kertész 2009) is a widely used benchmark with ground-truth clusters. In Figure 5(b), we generate five synthetic graphs composed of 1,000,000 vertices by varying the mixing ratio μ from 0.1 to 0.9. A larger μ implies that the number of edges crossing clusters increases, resulting in that being more difficult to detect intrinsic clusters. As can be seen, *PCon_de* consistently outperforms *SC* in terms of *NMI* and ϕ. Meanwhile, the quality of *PCon_de* decreases as μ increases, but it is always better than *SC*. Furthermore, we also generate four graphs with varying the number of vertices when fixing $\mu = 0.4$. Figure 5(c) shows a similar trend as Figure 5(b). As a consequence, these results imply that *PCon_de* approximates the ground-truth clusters more effectively than *SC*.

Conclusion

In this paper, we devise a *peeling*-based computing framework *PCon* for conductance-based graph clustering. We observe that most state-of-the-art algorithms can be reduced to *PCon*. Inspired by our framework, we first propose an efficient heuristic algorithm *PCon_core*, which adopts the degeneracy ordering to detect clusters. To improve the accuracy, we further propose a powerful *PCon_de* with near-constant approximation ratio, which achieve an important theoretical improvement over existing approaches such as *Fiedler* vector-based spectral clustering. Finally, extensive experiments on eleven datasets with six competitors demonstrate that the proposed algorithms can achieve $5 \sim 42$ times speedup with a high accuracy and $1.4 \sim 7.8$ times less memory than the state-of-the-art solutions.

Acknowledgments

References

Alon, N.; and Milman, V. D. 1985. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1): 73–88.

Andersen, R.; Chung, F. R. K.; and Lang, K. J. 2006. Local Graph Partitioning using PageRank Vectors. In *FOCS*, 475–486.

Arora, S.; Rao, S.; and Vazirani, U. V. 2004. Expander flows, geometric embeddings and graph partitioning. In Babai, L., ed., *STOC*. ACM.

Barabási, A.-L.; and Albert, R. 1999. Emergence of scaling in random networks. *science*, 286(5439): 509–512.

Batagelj, V.; and Zaversnik, M. 2003. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR*, cs.DS/0310049.

Belkin, M.; and Niyogi, P. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS*, 585–591.

Bianchi, F. M.; Grattarola, D.; and Alippi, C. 2020. Spectral Clustering with Graph Neural Networks for Graph Pooling. In *ICML*, 874–883.

Chang, L.; and Qin, L. 2019. Cohesive Subgraph Computation Over Large Sparse Graphs. In *ICDE*, 2068–2071.

Chawla, S.; Krauthgamer, R.; Kumar, R.; Rabani, Y.; and Sivakumar, D. 2006. On the Hardness of Approximating Multicut and Sparsest-Cut. *Comput. Complex.*, 15(2): 94–114.

Cilibrasi, R.; and Vitányi, P. M. B. 2005. Clustering by compression. *IEEE Trans. Inf. Theory*, 51(4): 1523–1545.

Erdos, P.; Rényi, A.; et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60.

Fiedler, M. 1973. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2): 298–305.

Fortunato, S. 2009. Community detection in graphs. *Physics Reports*, 486(3): 75–174.

Fountoulakis, K.; Wang, D.; and Yang, S. 2020. p-Norm Flow Diffusion for Local Graph Clustering. In *ICML*, volume 119, 3222–3232.

Hasenplaugh, W.; Kaler, T.; Schardl, T. B.; and Leiserson, C. E. 2014. Ordering heuristics for parallel graph coloring. In Blelloch, G. E.; and Sanders, P., eds., *SPAA*.

Holme, P.; and Kim, B. J. 2002. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2): 026107.

Kloster, K.; and Gleich, D. F. 2014. Heat kernel based community detection. In *KDD*, 1386–1395.

Lancichinetti, A.; Fortunato, S.; and Kertész, J. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New journal of physics*, 11(3): 033015.

Leighton, T.; and Rao, S. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6): 787–832.

Leskovec, J.; Lang, K. J.; and Mahoney, M. W. 2010. Empirical comparison of algorithms for network community detection. In *WWW*, 631–640.

Li, R.; Gao, S.; Qin, L.; Wang, G.; Yang, W.; and Yu, J. X. 2020. Ordering Heuristics for k-clique Listing. *Proc. VLDB Endow.*, 13(11): 2536–2548.

Newman, M. E. 2004. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6): 066133.

Orecchia, L.; and Zhu, Z. A. 2014. Flow-Based Algorithms for Local Graph Clustering. In Chekuri, C., ed., *SODA*, 1267–1286. SIAM.

Shi, J.; and Malik, J. 1997. Normalized Cuts and Image Segmentation. In *CVPR*, 731–737.

Shun, J.; Roosta-Khorasani, F.; Fountoulakis, K.; and Mahoney, M. W. 2016. Parallel Local Graph Clustering. *Proc. VLDB Endow.*, 9(12): 1041–1052.

Spielman, D. A.; and Teng, S. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, 81–90.

Tolliver, D.; and Miller, G. L. 2006. Graph Partitioning by Spectral Rounding: Applications in Image Segmentation and Clustering. In *CVPR*, 1053–1060.

Trevisan, L. 2017. Lecture notes on graph partitioning, expanders and spectral methods. *University of California, Berkeley, https://people. eecs. berkeley. edu/luca/books/expanders-2016. pdf*.

Veldt, N.; Gleich, D.; and Mahoney, M. 2016. A simple and strongly-local flow-based method for cut improvement. In *ICML*, 1938–1947.

Wang, D.; Fountoulakis, K.; Henzinger, M.; Mahoney, M. W.; and Rao, S. 2017. Capacity Releasing Diffusion for Speed and Locality. In *ICML*, volume 70, 3598–3607.

Watts, D. J.; and Strogatz, S. H. 1998. Collective dynamics of small-world networks. *nature*, 393(6684): 440–442.

Xu, X.; Yuruk, N.; Feng, Z.; and Schweiger, T. A. J. 2007. SCAN: a structural clustering algorithm for networks. In *KDD*, 824–833.

Yang, J.; and Leskovec, J. 2015. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.*, 42(1): 181–213.

Yang, R.; Xiao, X.; Wei, Z.; Bhowmick, S. S.; Zhao, J.; and Li, R. 2019. Efficient Estimation of Heat Kernel PageRank for Local Clustering. In *SIGMOD*, 1339–1356.

Zhu, Z. A.; Lattanzi, S.; and Mirrokni, V. S. 2013. A Local Algorithm for Finding Well-Connected Clusters. In *ICML*, 396–404.