Local Algorithms for Finding Densely Connected Clusters

Peter Macgregor [1] He Sun [1]

Abstract

Local graph clustering is an important algorithmic technique for analysing massive graphs, and has been widely applied in many research fields of data science. While the objective of most (local) graph clustering algorithms is to find a vertex set of low conductance, there has been a sequence of recent studies that highlight the importance of the inter-connection between clusters when analysing real-world datasets. Following this line of research, in this work we study local algorithms for finding a pair of vertex sets defined with respect to their inter-connection and their relationship with the rest of the graph. The key to our analysis is a new reduction technique that relates the structure of multiple sets to a *single* vertex set in the reduced graph. Among many potential applications, we show that our algorithms successfully recover densely connected clusters in the Interstate Disputes Dataset and the US Migration Dataset.

1. Introduction

Given an arbitrary vertex u of a graph $G = (V, E)$ as input, a local graph clustering algorithm finds some low-conductance set $S \subset V$ containing u, while the algorithm runs in time proportional to the size of the target cluster and independent of the size of the graph G. Because of the increasing size of available datasets, which makes centralised computation too expensive, local graph clustering has become an important learning technique for analysing a number of large-scale graphs and has been applied to solve many other learning and combinatorial optimisation problems (Andersen, 2010; Andersen et al., 2016; Wang et al., 2017; Yin et al., 2017; Takai et al., 2020; Fountoulakis et al., 2020; Liu & Gleich, 2020; Zhu et al., 2013).

[1]School of Informatics, University of Edinburgh, Edinburgh, United Kingdom. Correspondence to: Peter Macgregor <peter.macgregor@ed.ac.uk>, He Sun <h.sun@ed.ac.uk>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

1.1. Our Contribution

We study local graph clustering for learning the structure of clusters that are defined by their inter-connections, and present two local algorithms to achieve this objective in both undirected graphs and directed ones.

Our first result is a local algorithm for finding densely connected clusters in an undirected graph $G = (V, E)$: given any seed vertex u, our algorithm is designed to find *two* clusters L, R around u, which are densely connected to each other and are loosely connected to $V \setminus (L \cup R)$. The design of our algorithm is based on a new reduction that allows us to relate the connections between L, R and $V \setminus (L \cup R)$ to a *single* cluster in the resulting graph H, and a generalised analysis of Pagerank-based algorithms for local graph clustering. The significance of our designed algorithm is demonstrated by our experimental results on the Interstate Dispute Network from 1816 to 2010. By connecting two vertices (countries) with an undirected edge weighted according to the severity of their military disputes and using the USA as the seed vertex, our algorithm recovers two groups of countries that tend to have conflicts with each other, and shows how the two groups evolve over time. In particular, as shown in Figures 1(a)-(d), our algorithm not only identifies the changing roles of Russia, Japan, and eastern Europe in line with 20th century geopolitics, but also the reunification of east and west Germany around 1990.

We further study densely connected clusters in a *directed graph* (digraph). Specifically, given any vertex u in a digraph $G = (V, E)$ as input, our second local algorithm outputs two disjoint vertex sets L and R, such that (i) there are many edges *from* L *to* R, and (ii) $L \cup R$ is loosely connected to $V \setminus (L \cup R)$. The design of our algorithm is based on the following two techniques: (1) a new reduction that allows us to relate the edge weight from L to R, as well as the edge connections between $L \cup R$ and $V \setminus (L \cup R)$, to a *single* vertex set in the resulting *undirected* graph H; (2) a refined analysis of the `ESP`-based algorithm for local graph clustering. We show that our local algorithm is able to recover densely connected clusters in the US migration dataset, in which two vertex sets L, R defined as above could represent a higher-order migration trend. In particular, as shown in Figures 1(e)–(h), by using different counties as starting vertices, our algorithm uncovers refined and more

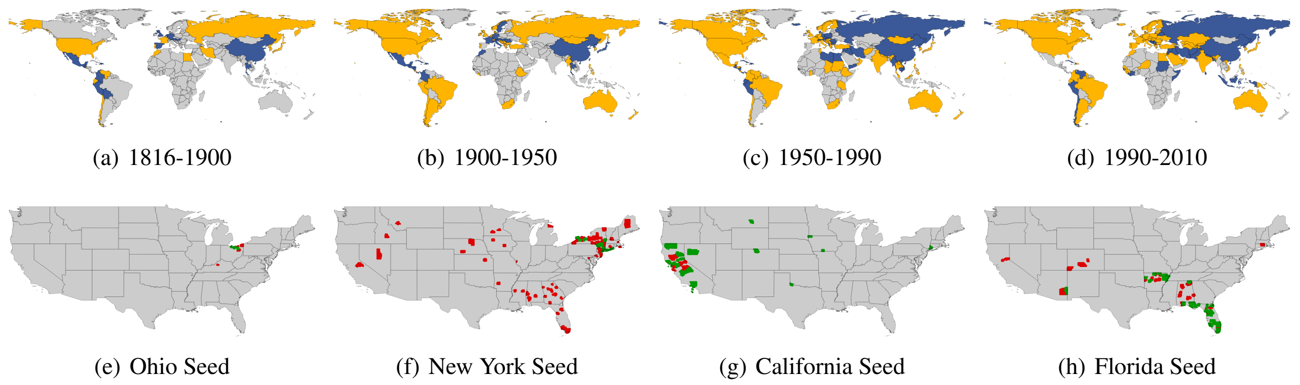

(a) 1816-1900
(b) 1900-1950
(c) 1950-1990
(d) 1990-2010
(e) Ohio Seed
(f) New York Seed
(g) California Seed
(h) Florida Seed

Figure 1. (a)-(d) Clusters found by `LocBipartDC` on the Interstate Dispute Network, using the USA as the seed vertex. In each case, countries in the yellow cluster tend to enter conflicts with countries in the blue cluster and vice versa. (e)-(h) Clusters found by `EvoCutDirected` in the US migration network. There is a significant migration trend from the red counties to the green counties.

localised migration patterns than the previous work on the same dataset (Cucuringu et al., 2020). To the best of our knowledge, our work represents the first local clustering algorithm that achieves a similar goal.

1.2. Related Work

Li and Peng (2013) study the same problem for undirected graphs, and present a random walk based algorithm. Andersen (2010) studies a similar problem for undirected graphs under a different objective function, and our algorithm's runtime significantly improves the one in (Andersen, 2010). There is some recent work on local clustering for hypergraphs (Takai et al., 2020), and algorithms for finding higher-order structures of graphs based on network motifs both centrally (Benson et al., 2015; 2016) and locally (Yin et al., 2017). These algorithms are designed to find different types of clusters, and cannot be directly compared with ours. Our problem is related to the problem of finding clusters in *disassortative* networks studied in (Moore et al., 2011; Pei et al., 2019; Zhu et al., 2020), although these works consider semi-supervised, global methods while ours is unsupervised and local. There are also recent studies which find clusters with a specific structure in the non-local setting (Cucuringu et al., 2020; Laenen & Sun, 2020). Our current work shows that such clusters can be learned locally via our presented new techniques.

2. Preliminaries

Notation. For any undirected and unweighted graph $G = (V_G, E_G)$ with n vertices and m edges, the degree of any vertex v is denoted by $\deg_G(v)$, and the set of neighbours of v is $N_G(v)$. For any $S \subseteq V$, its volume is

$$\mathrm{vol}_G(S) \triangleq \sum_{v \in S} \deg(v),$$

its boundary is $\partial_G(S) \triangleq \{(u,v) \in E : u \in S \text{ and } v \notin S\}$, and its conductance is

$$\Phi_G(S) \triangleq \frac{|\partial_G(S)|}{\min\{\mathrm{vol}_G(S), \mathrm{vol}_G(V \setminus S)\}}.$$

For disjoint $S_1, S_2 \subset V_G$, $e(S_1, S_2)$ is the number of edges between S_1 and S_2. When $G = (V_G, E_G)$ is a digraph, for any $u \in V_G$ we use $\deg_{\mathrm{out}}(u)$ and $\deg_{\mathrm{in}}(u)$ to express the number of edges with u as the tail or the head, respectively. For any $S \subset V_G$, we define $\mathrm{vol}_{\mathrm{out}}(S) = \sum_{u \in S} \deg_{\mathrm{out}}(u)$, and $\mathrm{vol}_{\mathrm{in}}(S) = \sum_{u \in S} \deg_{\mathrm{in}}(u)$.

For undirected graphs, we use D_G to denote the $n \times n$ diagonal matrix with $(D_G)_{v,v} = \deg_G(v)$ for any vertex $v \in V$, and we use A_G to represent the adjacency matrix of G defined by $(A_G)_{u,v} = 1$ if $\{u,v\} \in E_G$, and $(A_G)_{u,v} = 0$ otherwise. The lazy random walk matrix of G is $W_G = (1/2) \cdot (I + D_G^{-1} A_G)$. For any set $S \subset V_G$, χ_S is the indicator vector of S, i.e., $\chi_S(v) = 1$ if $v \in S$, and $\chi_S(v) = 0$ otherwise. If the set consists of a single vertex v, we write χ_v instead of $\chi_{\{v\}}$. Sometimes we drop the subscript G when the underlying graph is clear from the context. For any vectors $x, y \in \mathbb{R}^n$, we write $x \preceq y$ if it holds for any v that $x(v) \leq y(v)$. For any operators $f, g : \mathbb{R}^n \to \mathbb{R}^n$, we define $f \circ g : \mathbb{R}^n \to \mathbb{R}^n$ by $f \circ g(v) \triangleq f(g(v))$ for any v. For any vector p, we define the support of p to be $\mathrm{supp}(p) = \{u : p(u) \neq 0\}$. The sweep sets of p are defined by (1) ordering all the vertices such that $\frac{p(v_1)}{\deg(v_1)} \geq \frac{p(v_2)}{\deg(v_2)} \geq \ldots \geq \frac{p(v_n)}{\deg(v_n)}$, and (2) constructing $S_j^p = \{v_1, \ldots, v_j\}$ for $1 \leq j \leq n$. Throughout this paper, we will consider vectors to be row vectors, so the random walk update step for a distribution p is written as pW. For ease of presentation we consider only unweighted graphs; however, our analysis can be easily generalised to the weighted case. All the omitted and technical details can be found in the full version (arXiv:2106.05245).

Pagerank. Given an underlying graph G with the lazy random walk matrix W, the personalised Pagerank vector $\mathrm{pr}(\alpha, s)$ is defined to be the unique solution of the equation

$$\mathrm{pr}(\alpha, s) = \alpha s + (1-\alpha)\mathrm{pr}(\alpha, s)W, \qquad (1)$$

where $s \in \mathbb{R}^n_{\geq 0}$ is a starting vector and $\alpha \in (0,1]$ is called the teleport probability. Andersen et al. (2006) show that the personalised Pagerank vector can be written as $\mathrm{pr}(\alpha, s) = \alpha \sum_{t=0}^{\infty}(1-\alpha)^t s W^t$. Therefore, we could study $\mathrm{pr}(\alpha, s)$ through the following random process: pick some integer $t \in \mathbb{Z}_{\geq 0}$ with probability $\alpha(1-\alpha)^t$, and perform a t-step lazy random walk, where the starting vertex of the random walk is picked according to s. Then, $\mathrm{pr}(\alpha, s)$ describes the probability of reaching each vertex in this process.

Computing an exact Pagerank vector $\mathrm{pr}(\alpha, \chi_v)$ is equivalent to computing the stationary distribution of a Markov chain on the vertex set V which has a time complexity of $\Omega(n)$. However, since the probability mass of a personalised Pagerank vector is concentrated around some starting vertex, it is possible to compute a good approximation of the Pagerank vector in a local way. Andersen et al. (2006) introduce the approximate Pagerank, which will be used in our analysis.

Definition 1. *A vector $p = \mathrm{apr}(\alpha, s, r)$ is an approximate Pagerank vector if $p + \mathrm{pr}(\alpha, r) = \mathrm{pr}(\alpha, s)$. The vector r is called the residual vector.*

The evolving set process. The evolving set process (ESP) is a Markov chain whose states are sets of vertices $S_i \subseteq V$. Given a state S_i, the next state S_{i+1} is determined by the following process: (1) choose $t \in [0,1]$ uniformly at random; (2) let $S_{i+1} = \{v \in V | \chi_v W \chi_{S_i}^{\intercal} \geq t\}$. The volume-biased ESP is a variant used to ensure that the Markov chain absorbs in the state V rather than $\emptyset$. Andersen and Peres (2009) give a local algorithm for undirected graph clustering using the volume-biased ESP. In particular, they give an algorithm `GenerateSample`(u, T) which samples the T-th element from the volume-biased ESP with $S_0 = \{u\}$.

3. The Algorithm for Undirected Graphs

Now we present a local algorithm for finding two clusters in an undirected graph with a dense cut between them. To formalise this notion, for any undirected graph $G = (V, E)$ and disjoint $L, R \subset V$, we follow Trevisan (2012) and define the *bipartiteness* ratio as

$$\beta(L, R) \triangleq 1 - \frac{2e(L,R)}{\mathrm{vol}(L \cup R)}.$$

Notice that a low $\beta(L, R)$ value means that there is a dense cut between L and R, and there is a sparse cut between $L \cup R$ and $V \setminus (L \cup R)$. In particular, $\beta(L, R) = 0$ implies that (L, R) forms a bipartite and connected component of G. We will describe a local algorithm for finding almost-bipartite sets L and R with a low value of $\beta(L, R)$.

3.1. The Reduction by Double Cover

The design of most local algorithms for finding a target set $S \subset V$ of low conductance is based on analysing the behaviour of random walks starting from vertices in S. In particular, when the conductance $\Phi_G(S)$ is low, a random walk starting from most vertices in S will leave S with low probability. However, for our setting, the target is a pair of sets L, R with many connections between them. As such, a random walk starting in either L or R is very likely to leave the starting set. To address this, we introduce a novel technique based on the double cover of G to reduce the problem of finding two sets of high conductance to the problem of finding one of *low* conductance.

Formally, for any undirected graph $G = (V_G, E_G)$, its double cover is the bipartite graph $H = (V_H, E_H)$ defined as follows: (1) every vertex $v \in V_G$ has two corresponding vertices $v_1, v_2 \in V_H$; (2) for every edge $\{u, v\} \in E_G$, there are edges $\{u_1, v_2\}$ and $\{u_2, v_1\}$ in E_H. See Figure 2 for an illustration.

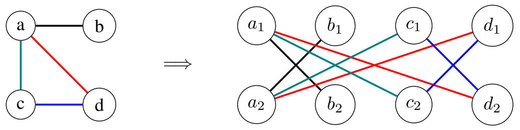


Figure 2. An example of the construction of the double cover.

Now we present a tight connection between the value of $\beta(L, R)$ for any disjoint sets $L, R \subset V_G$ and the conductance of a *single* set in the double cover of G. To this end, for any $S \subset V_G$, we define $S_1 \subset V_H$ and $S_2 \subset V_H$ by $S_1 \triangleq \{v_1 \mid v \in S\}$ and $S_2 \triangleq \{v_2 \mid v \in S\}$. We formalise the connection in the following lemma.

Lemma 1. *Let G be an undirected graph, $S \subset V$ with partitioning (L, R), and H be the double cover of G. Then, it holds that $\Phi_H(L_1 \cup R_2) = \beta_G(L, R)$.*

Next we look at the other direction of this correspondence. Specifically, given any $S \subset V_H$ in the double cover of a graph G, we would like to find two disjoint sets $L \subset V_G$ and $R \subset V_G$ such that $\beta_G(L, R) = \Phi_H(S)$. However, such a connection does not hold in general. To overcome this, we restrict our attention to those subsets of V_H which can be unambiguously interpreted as two disjoint sets in V_G.

Definition 2. *We call $S \subset V_H$* ***simple*** *if $|\{v_1, v_2\} \cap S| \leq 1$ holds for all $v \in V_G$.*

Lemma 2. *For any simple set $S \subset V_H$, let $L = \{u : u_1 \in S\}$ and $R = \{u : u_2 \in S\}$. Then, $\beta_G(L, R) = \Phi_H(S)$.*

3.2. Design of the Algorithm

So far we have shown that the problem of finding densely connected sets $L, R \subset V_G$ can be reduced to finding $S \subset V_H$ of low conductance in the double cover H, and this reduction raises the natural question of whether existing local algorithms can be directly employed to find L and R in G. However, this is not the case: even though a set $S \subset V_H$ returned by most local algorithms is guaranteed to have low conductance, vertices of G could be included in S twice, and as such S will not necessarily give us disjoint sets $L, R \subset V_G$ with low value of $\beta(L, R)$. See Figure 3 for an illustration.

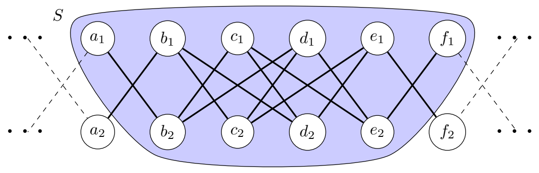


Figure 3. The indicated vertex set $S \subset V_H$ has low conductance. However, since S contains many pairs of vertices which correspond to the same vertex in G, S gives us little information for finding disjoint $L, R \subset V_G$ with a low $\beta(L, R)$ value.

The simplify operator. To take the example shown in Figure 3 into account, our objective is to design a local algorithm for finding some set $S \subset V_H$ of low conductance which is also simple. To ensure this, we introduce the simplify operator, and analyse its properties.

Definition 3 (Simplify operator)**.** *Let H be the double cover of G, where the two corresponding vertices of any $u \in V_G$ are defined as $u_1, u_2 \in V_H$. Then, for any $p \in \mathbb{R}^{2n}_{\geq 0}$, the simplify operator is a function $\sigma : \mathbb{R}^{2n}_{\geq 0} \to \mathbb{R}^{2n}_{\geq 0}$ defined by*

$$(\sigma \circ p)(u_1) \triangleq \max(0, p(u_1) - p(u_2)),$$
$$(\sigma \circ p)(u_2) \triangleq \max(0, p(u_2) - p(u_1))$$

for every $u \in V_G$.

Notice that, for any vector p and any $u \in V_G$, at most one of u_1 and u_2 is in the support of $\sigma \circ p$; hence, the support of $\sigma \circ p$ is always simple. To explain the meaning of σ, for a vector p one could view $p(u_1)$ as our "confidence" that $u \in L$, and $p(u_2)$ as our "confidence" that $u \in R$. Hence, when $p(u_1) \approx p(u_2)$, both $(\sigma \circ p)(u_1)$ and $(\sigma \circ p)(u_2)$ are small which captures the fact that we would not prefer to include u in either L or R. On the other hand, when $p(u_1) \gg p(u_2)$, we have $(\sigma \circ p)(u_1) \approx p(u_1)$, which captures our confidence that u should belong to L. The following lemma summaries some key properties of σ.

Lemma 3. *The following holds for the σ-operator:*

- *$\sigma \circ (c \cdot p) = c \cdot (\sigma \circ p)$ for $p \in \mathbb{R}^{2n}_{\geq 0}$ and any $c \in \mathbb{R}_{\geq 0}$;*
- *$\sigma \circ (a + b) \preceq \sigma \circ a + \sigma \circ b$ for $a, b \in \mathbb{R}^{2n}_{\geq 0}$;*
- *$\sigma \circ (pW) \preceq (\sigma \circ p)W$ for $p \in \mathbb{R}^{2n}_{\geq 0}$.*

While these three properties will all be used in our analysis, the third is of particular importance: it implies that, if p is the probability distribution of a random walk in H, applying σ before taking a one-step random walk would never result in lower probability mass than applying σ after taking a one-step random walk. This means that no probability mass would be lost when the σ-operator is applied between every step of a random walk, in comparison with applying σ at the end of an entire random walk process.

Description of the algorithm. Our proposed algorithm is conceptually simple: every vertex u of the input graph G maintains two copies u_1, u_2 of itself, and these two "virtual" vertices are used to simulate u's corresponding vertices in the double cover H of G. Then, as the neighbours of u_1 and u_2 in H are entirely determined by u's neighbours in G and can be constructed locally, a random walk process in H will be simulated in G. This will allow us to apply a local algorithm similar to the one by Anderson et al. (2006) on this "virtual" graph H. Finally, since all the required information about $u_1, u_2 \in V_H$ is maintained by $u \in V_G$, the σ-operator will be applied locally.

The formal description of our algorithm is given in Algorithm 1, which invokes Algorithm 2 as the key component to compute $\mathrm{apr}_H(\alpha, \chi_{u_1}, r)$. Specifically, Algorithm 2 maintains, for every vertex $u \in G$, tuples $(p(u_1), p(u_2))$ and $(r(u_1), r(u_2))$ to keep track of the values of p and r in G's double cover. For a given vertex u, the entries in these tuples are expressed by $p_1(u), p_2(u), r_1(u)$, and $r_2(u)$ respectively. Every `dcpush` operation (Algorithm 3) preserves the invariant $p + \mathrm{pr}_H(\alpha, r) = \mathrm{pr}_H(\alpha, \chi_{v_1})$, which ensures that the final output of Algorithm 2 is an approximate Pagerank vector. We remark that, although the presentation of the `ApproximatePagerankDC` procedure is similar to the one in Andersen et al. (2006), in our `dcpush` procedure the update of the residual vector r is slightly more involved: specifically, for every vertex $u \in V_G$, both $r_1(u)$ and $r_2(u)$ are needed in order to update $r_1(u)$ (or $r_2(u)$). That is one of the reasons that the performance of the algorithm in Andersen et al. (2006) cannot be directly applied for our algorithm, and a more technical analysis, some of which is parallel to theirs, is needed in order to analyse the correctness and performance of our algorithm.

3.3. Analysis of the Algorithm

To prove the correctness of our algorithm, we will show two complementary facts which we state informally here:

1. If there is a simple set $S \subset V_H$ with low conductance, then for most $u_1 \in S$, the simplified approximate Pagerank vector $p = \sigma \circ \mathrm{apr}(\alpha, \chi_{u_1}, r)$ will have a lot of probability mass on a small set of vertices.
2. If $p = \sigma \circ \mathrm{apr}(\alpha, \chi_{u_1}, r)$ contains a lot of probability mass on some small set of vertices, then there is a sweep set of p with low conductance.

Algorithm 1 `LocBipartDC`

```
    Input: A graph $G$, starting vertex $u$, target volume $\gamma$, and target bipartiteness $\beta$
    Output: Two disjoint sets $L$ and $R$
    Set $\alpha = \frac{\beta^2}{378}$, and $\epsilon = \frac{1}{20\gamma}$
    Compute $p' = \texttt{ApproximatePagerankDC}(u, \alpha, \epsilon)$
    Compute $p = \sigma \circ p'$
    for $j \in [1, |\mathrm{supp}(p)|]$ do
      if $\Phi(S_j^p) \leq \beta$ then
        Set $L = \{u : u_1 \in S_j^p\}$, and $R = \{u : u_2 \in S_j^p\}$
        return $(L, R)$
      end if
    end for
```

Algorithm 2 `ApproximatePagerankDC`

```
    Input: Starting vertex $v$, parameters $\alpha$ and $\epsilon$
    Output: Approximate Pagerank vector $\mathrm{apr}_H(\alpha, \chi_{v_1}, r)$
    Set $p_1 = p_2 = r_2 = \mathbf{0}$; set $r_1 = \chi_v$
    while $\max_{(u,i) \in V \times \{1,2\}} \frac{r_i(u)}{\deg(u)} \geq \epsilon$ do
      Choose any $u$ and $i \in \{1, 2\}$ such that $\frac{r_i(u)}{\deg(u)} \geq \epsilon$
      $(p_1, p_2, r_1, r_2) = \texttt{dcpush}(\alpha, (u, i), p_1, p_2, r_1, r_2)$
    end while
    return $p = [p_1, p_2]$
```

As we have shown in Section 3.1, there is a direct correspondence between densely connected sets in G, and low-conductance and simple sets in H. This means that the two facts above are exactly what we need to prove that Algorithm 1 can find densely connected sets in G.

We will first show in Lemma 4 how the σ-operator affects some standard mixing properties of Pagerank vectors in order to establish the first fact promised above. This lemma relies on the fact that $S \subset V_G$ corresponds to a *simple* set in V_H. This allows us to apply the σ-operator to the approximate Pagerank vector $\mathrm{apr}_H(\alpha, \chi_{u_1}, r)$ while preserving a large probability mass on the target set.

Lemma 4. *For any set $S \subset V_G$ with partitioning (L, R) and any constant $\alpha \in [0, 1]$, there is a subset $S_\alpha \subseteq S$ with $\mathrm{vol}(S_\alpha) \geq \mathrm{vol}(S)/2$ such that, for any vertex $v \in S_\alpha$, the simplified approximate Pagerank on the double cover $p = \sigma \circ (\mathrm{apr}_H(\alpha, \chi_{v_1}, r))$ satisfies*

$$p(L_1 \cup R_2) \geq 1 - \frac{2\beta(L, R)}{\alpha} - 2\mathrm{vol}(S) \max_{u \in V} \frac{r(u)}{\deg(u)}.$$

To prove the second fact, we show as an intermediate lemma that the value of $p(u_1)$ can be bounded with respect to its value after taking a step of the random walk: $pW(u_1)$.

Lemma 5. *Let G be a graph with double cover H, and $\mathrm{apr}(\alpha, s, r)$ be the approximate Pagerank vector defined with respect to H. Then, $p = \sigma \circ (\mathrm{apr}(\alpha, s, r))$ satisfies*

Algorithm 3 `dcpush`

```
    Input: $\alpha, (u, i), p_1, p_2, r_1, r_2$
    Output: $(p'_1, p'_2, r'_1, r'_2)$
    Set $(p'_1, p'_2, r'_1, r'_2) = (p_1, p_2, r_1, r_2)$
    Set $p'_i(u) = p_i(u) + \alpha r_i(u)$; $r'_i(u) = (1 - \alpha)\frac{r_i(u)}{2}$
    for $v \in N_G(u)$ do
      Set $r'_{3-i}(v) = r_{3-i}(v) + (1 - \alpha)\frac{r_i(u)}{2\deg(u)}$
    end for
    return $(p'_1, p'_2, r'_1, r'_2)$
```

that $p(u_1) \leq \alpha\left(s(u_1) + r(u_2)\right) + (1 - \alpha)(pW)(u_1)$, and $p(u_2) \leq \alpha\left(s(u_2) + r(u_1)\right) + (1 - \alpha)(pW)(u_2)$ for any $u \in V_G$.

Notice that applying the σ-operator for any vertex u_1 introduces a new dependency on the value of the residual vector r at u_2. This subtle observation demonstrates the additional complexity introduced by the σ-operator when compared with previous analysis of Pagerank-based local algorithms (Andersen et al., 2006). Taking account of the σ-operator, we further analyse the Lovász-Simonovits curve defined by p, which is a common technique in the analysis of random walks on graphs (Lovász & Simonovits, 1990): we show that if there is a set S with a large value of $p(S)$, there must be a sweep set S_j^p with small conductance.

Lemma 6. *Let G be a graph with double cover H, and let $p = \sigma \circ (\mathrm{apr}_H(\alpha, s, r))$ such that $\max_{u \in V_H} \frac{r(u)}{d(u)} \leq \epsilon$. If there is a set of vertices $S \subset V_H$ and a constant δ such that $p(S) - \frac{\mathrm{vol}(S)}{\mathrm{vol}(V_H)} \geq \delta$, then there is some $j \in [1, |\mathrm{supp}(p)|]$ such that $\Phi_H(S_j^p) < 6\sqrt{(1 + \epsilon\mathrm{vol}(S))\alpha \ln(\frac{4}{\delta})/\delta}$.*

We have now shown the two facts promised at the beginning of this subsection. Putting these together, if there is a simple set $S \subset V_H$ with low conductance then we can find a sweep set of $\sigma \circ \mathrm{apr}(\alpha, s, r)$ with low conductance. By the reduction from almost-bipartite sets in G to low-conductance simple sets in H our target set corresponds to a simple set $S \subset V_H$ which leads to Algorithm 1 for finding almost-bipartite sets. Our result is summarised as follows.

Theorem 1. *Let G be an n-vertex undirected graph, and $L, R \subset V_G$ be disjoint sets such that $\beta(L, R) \leq \beta$ and $\mathrm{vol}(L \cup R) \leq \gamma$. Then, there is a set $C \subseteq L \cup R$ with $\mathrm{vol}(C) \geq \mathrm{vol}(L \cup R)/2$ such that, for any $v \in C$, `LocBipartDC`$\left(G, v, \gamma, \sqrt{7560\beta}\right)$ returns (L', R') with $\beta(L', R') = O\left(\sqrt{\beta}\right)$ and $\mathrm{vol}(L' \cup R') = O\left(\beta^{-1}\gamma\right)$. Moreover, the algorithm has running time $O\left(\beta^{-1}\gamma \log n\right)$.*

The quadratic approximation guarantee in Theorem 1 matches the state-of-the-art local algorithm for finding a single set with low conductance (Andersen et al., 2016). Furthermore, our result presents a significant improvement over the previous state-of-the-art by Li and Peng (2013), whose design is based on an entirely different technique than ours. For any $\epsilon \in [0, 1/2]$, their algorithm runs in time $O\left(\epsilon^2\beta^{-2}\gamma^{1+\epsilon}\log^3\gamma\right)$ and returns a set with volume $O\left(\gamma^{1+\epsilon}\right)$ and bipartiteness ratio $O\left(\sqrt{\beta/\epsilon}\right)$. In particular, their algorithm requires much higher time complexity in order to guarantee the same bipartiteness ratio $O\left(\sqrt{\beta}\right)$.

4. The Algorithm for Digraphs

We now turn our attention to local algorithms for finding densely connected clusters in digraphs. In comparison with undirected graphs, we are interested in finding disjoint $L, R \subset V$ of some digraph $G = (V, E)$ such that most of the edges adjacent to $L \cup R$ are *from L to R*. To formalise this, we define the *flow ratio* from L to R as

$$F(L, R) \triangleq 1 - \frac{2e(L, R)}{\mathrm{vol}_{\mathrm{out}}(L) + \mathrm{vol}_{\mathrm{in}}(R)},$$

where $e(L, R)$ is the number of directed edges from L to R. Notice that we take not only edge densities but also edge directions into account: a low $F(L, R)$-value tells us that almost all edges with their tail in L have their head in R, and conversely almost all edges with their head in R have their tail in L. One could also see $F(L, R)$ as a generalisation of $\beta(L, R)$. In particular, if we view an undirected graph as a digraph by replacing each edge with two directed edges, then $\beta(L, R) = F(L, R)$. In this section, we will present a local algorithm for finding such vertex sets in a digraph, and analyse the algorithm's performance.

4.1. The Reduction by Semi-Double Cover

Given a digraph $G = (V_G, E_G)$, we construct its *semi-double cover* $H = (V_H, E_H)$ as follows: (1) every vertex $v \in V_G$ has two corresponding vertices $v_1, v_2 \in V_H$; (2) for every edge $(u, v) \in E_G$, we add the edge $\{u_1, v_2\}$ in E_H. [1] It is worth comparing this reduction with the one for undirected graphs:

- For undirected graphs, we apply the standard double cover and every undirected edge in G corresponds to two edges in the double cover H;
- For digraphs, every directed edge in G corresponds to *one* undirected edge in H. This *asymmetry* would allow us to "recover" the direction of any edge in G.

We follow the use of S_1, S_2 from Section 3: for any $S \subset V_G$, we define $S_1 \subset V_H$ and $S_2 \subset V_H$ by $S_1 \triangleq \{v_1 \mid v \in S\}$ and $S_2 \triangleq \{v_2 \mid v \in S\}$. The lemma below shows the connection between the value of $F_G(L, R)$ for any L, R and $\Phi_H(L_1 \cup R_2)$.

Lemma 7. *Let G be a digraph with semi-double cover H. Then, it holds for any $L, R \subset V_G$ that $F_G(L, R) = \Phi_H(L_1 \cup R_2)$. Similarly, for any simple set $S \subset V_H$, let $L = \{u : u_1 \in S\}$ and $R = \{u : u_2 \in S\}$. Then, it holds that $F_G(L, R) = \Phi_H(S)$.*

[1]We remark that this reduction was also used by Anderson (2010) for finding dense components in a digraph.

4.2. Design and Analysis of the Algorithm

Our presented algorithm is a modification of the algorithm by Andersen and Peres (2009). Given a digraph G as input, our algorithm simulates the volume-biased ESP on G's semi-double cover H. Notice that the graph H can be constructed locally in the same way as the local construction of the double cover. However, as the output set S of an ESP-based algorithm is not necessarily simple, our algorithm only returns vertices $u \in V_G$ in which *exactly* one of u_1 and u_2 is included in S. The key procedure for our algorithm is given in Algorithm 4, in which the `GenerateSample` procedure is the one described at the end of Section 2.

Algorithm 4 `EvoCutDirected (ECD)`

```
Input: Starting vertex $u$, $i \in \{1, 2\}$, target flow ratio $\phi$
Output: A pair of sets $L, R \subset V_G$
Set $T = \lfloor (100\phi^{\frac{2}{3}})^{-1} \rfloor$.
Compute $S = \mathtt{GenerateSample}_H(u_i, T)$
Let $L = \{u \in V_G : u_1 \in S$ and $u_2 \notin S\}$
Let $R = \{u \in V_G : u_2 \in S$ and $u_1 \notin S\}$
return $L$ and $R$
```

Notice that in our constructed graph H, $\Phi(L_1 \cup R_2) \leq \phi$ does not imply that $\Phi(L_2 \cup R_1) \leq \phi$. Due to this asymmetry, Algorithm 4 takes a parameter $i \in \{1, 2\}$ to indicate whether the starting vertex is in L or R. If it is not known whether u is in L or R, two copies can be run in parallel, one with $i = 1$ and the other with $i = 2$. Once one of them terminates with the performance guaranteed in Theorem 2, the other can be terminated. Hence, we can always assume that it is known whether the starting vertex u is in L or R.

Now we sketch the analysis of the algorithm. Notice that, since the evolving set process gives us an arbitrary set on the semi-double cover, in Algorithm 4 we convert this into a simple set by removing any vertices u where $u_1 \in S$ and $u_2 \in S$. The following definition allows us to discuss sets which are close to being simple.

Definition 4 (ϵ-simple set). *For any set $S \subset V_H$, let $P = \{u_1, u_2 : u \in V_G, u_1 \in S$ and $u_2 \in S\}$. We call set S ϵ-simple if it holds that $\frac{\mathrm{vol}(P)}{\mathrm{vol}(S)} \leq \epsilon$.*

The notion of ϵ-simple sets measures the ratio of vertices in which both u_1 and u_2 are in S. In particular, any simple set defined in Definition 2 is 0-simple. We show that, for any ϵ-simple set $S \subset V_H$, one can construct a simple set S' such that $\Phi(S') \leq \frac{1}{1-\epsilon} \cdot (\Phi(S) + \epsilon)$. Therefore, in order to guarantee that $\Phi(S')$ is small, we need to construct S such that $\Phi(S)$ is small and S is ϵ-simple for small ϵ. Because of this, our presented algorithm uses a lower value of T than the algorithm in Andersen and Peres (2009); this allows us to better control $\mathrm{vol}(S)$ at the cost of a slightly worse approximation guarantee. Our algorithm's performance is summarised in Theorem 2.

Theorem 2. *Let G be an n-vertex digraph, and $L, R \subset V_G$ be disjoint sets such that $F(L,R) \leq \phi$ and $\mathrm{vol}(L \cup R) \leq \gamma$. There is a set $C \subseteq L \cup R$ with $\mathrm{vol}(C) \geq \mathrm{vol}(L \cup R)/2$ such that, for any $v \in C$ and some $i \in \{1,2\}$, `EvoCutDirected`(G, v, i, ϕ) returns (L', R') such that $F(L', R') = O\left(\phi^{\frac{1}{3}} \log^{\frac{1}{2}} n\right)$ and $\mathrm{vol}(L' \cup R') = O\left((1-\phi^{\frac{1}{3}})^{-1}\gamma\right)$. Moreover, the algorithm has running time $O\left(\phi^{-\frac{1}{2}}\gamma \log^{\frac{3}{2}} n\right)$.*

To the best of our knowledge, this is the first local algorithm for digraphs that approximates a pair of densely connected clusters, and demonstrates that finding such a pair appears to be much easier than finding a low-conductance set in a digraph; in particular, existing local algorithms for finding a low-conductance set require the stationary distribution of the random walk in the digraph (Andersen & Chung, 2007), the sublinear-time computation of which is unknown (Cohen et al., 2017). However, knowledge of the stationary distribution is not needed for our algorithm.

Further Discussion. It is important to note that the semi-double cover construction is able to handle directed graphs which contain edges between two vertices u and v in both directions. In other words, the adjacency matrix of the digraph need not be skew-symmetric. This is an advantage of our approach over previous methods (e.g., (Cucuringu et al., 2020)), and it would be a meaningful research direction to identify the benefit this gives our developed reduction.

It is also insightful to discuss why Algorithm 1 cannot be applied for digraphs, although the input digraph is translated into an *undirected* graph by our reduction. This is because, when translating a digraph into a bipartite undirected graph, the third property of the σ-operator in Lemma 3 no longer holds, since the existence of the edge $\{u_1, v_2\} \in E_H$ does not necessarily imply that $\{u_2, v_1\} \in E_H$. Indeed, Figure 4 gives a counterexample in which $(\sigma \circ (pW))(u) \not\preceq ((\sigma \circ p)W)(u)$ for some u. This means that the typical analysis of a Pagerank vector with the Lovász-Simonovitz curve cannot be applied anymore. In our point of view, constructing some operator similar to our σ-operator and applying this operator to design a Pagerank-based local algorithm for digraphs is a very interesting open question, and may help to close the gap in the approximation guarantee between the undirected and directed cases.

In addition, we underline that one cannot apply the tighter analysis of the ESP process given by Andersen et al. (2016) to our algorithm. The key to their analysis is an improved

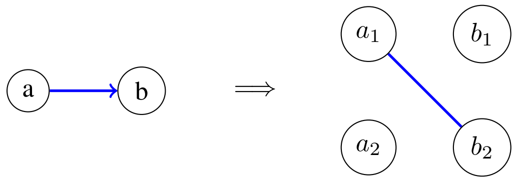


Figure 4. Consider the digraph and its semi-double cover above. Suppose $p(a_1) = p(a_2) = 0.5$ and $p(b_1) = p(b_2) = 0$. It is straightforward to check that $(\sigma \circ (pW))(b_2) = 0.25$ and $((\sigma \circ p)W)(b_2) = 0$.

bound on the probability that a random walk escapes from the target cluster. In order to take advantage of this, they use a larger value of T in the algorithm which relaxes the guarantee on the volume of the output set. Since our analysis relies on a very tight guarantee on the overlap of the output set with the target set, we cannot use their improvement in our setting.

5. Experiments

In this section we evaluate the performance of our proposed algorithms on both synthetic and real-world data sets. For undirected graphs, we compare the performance of our algorithm against the previous state-of-the-art (Li & Peng, 2013), referred to as `LP`, through the synthetic dataset with various parameters and apply the real-world dataset to demonstrate the significance of our algorithm. For directed graphs, we compare the performance of our algorithm with the state-of-the-art *non-local* algorithm since, to the best of our knowledge, our local algorithm for digraphs is the first such algorithm in the literature. All experiments were performed on a Lenovo Yoga 2 Pro with an Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz processor and 8GB of RAM. We include additional discussion in the full version. Our code can be downloaded from https://github.com/pmacg/local-densely-connected-clusters.

5.1. Results for Undirected Graphs

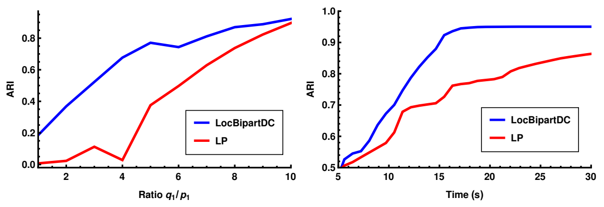
Synthetic Dataset. We first compare the performance of our algorithm with the previously best one on graphs generated from the stochastic block model (SBM). Specifically, we assume that the graph has $k = 3$ clusters $\{C_j\}_{j=1}^3$, and the number of vertices in each cluster, denoted by n_1, n_2 and n_3 respectively, satisfy $n_1 = n_2 = 0.1n_3$. Moreover, any pair of vertices $u \in C_i$ and $v \in C_j$ is connected with probability $P_{i,j}$. We assume that $P_{1,1} = P_{2,2} = p_1$, $P_{3,3} = p_2$, $P_{1,2} = q_1$, and $P_{1,3} = P_{2,3} = q_2$. Throughout our experiments, we maintain the ratios $p_2 = 2p_1$ and $q_2 = 0.1p_1$, leaving the parameters n_1, p_1 and q_1 free. Notice that the different values of q_1 and q_2 guarantee that C_1 and C_2 are the ones optimising the β-value, which is why our proposed model is slightly more involved than the standard SBM.

Table 1. Full comparison between Algorithm 1 (`LocBipartDC`) and the previous state-of-the-art (Li & Peng, 2013). For clarity we report the target bipartiteness $\beta = \beta(C_1, C_2)$ and target volume $\gamma = \mathrm{vol}(C_1 \cup C_2)$ along with the SBM parameters.

Input graph parameters	Algorithm	Runtime	β-value	ARI	Misclassified Ratio
$n_1 = 1{,}000$, $p_1 = 0.001$, $q_1 = 0.018$	LocBipartDC	**0.09**	**0.154**	**0.968**	**0.073**
$\beta \approx 0.1$, $\gamma \approx 40{,}000$	LP	0.146	0.202	0.909	0.138
$n_1 = 10{,}000$, $p_1 = 0.0001$, $q_1 = 0.0018$	LocBipartDC	**0.992**	**0.215**	**0.940**	**0.145**
$\beta \approx 0.1$, $\gamma \approx 400{,}000$	LP	1.327	0.297	0.857	0.256
$n_1 = 100{,}000$, $p_1 = 0.00001$, $q_1 = 0.00018$	LocBipartDC	**19.585**	**0.250**	**0.950**	**0.166**
$\beta \approx 0.1$, $\gamma \approx 4{,}000{,}000$	LP	30.285	0.300	0.865	0.225
$n_1 = 1{,}000$, $p_1 = 0.004$, $q_1 = 0.012$	LocBipartDC	**1.249**	**0.506**	**0.503**	**0.763**
$\beta \approx 0.4$, $\gamma \approx 40{,}000$	LP	1.329	0.597	0.445	0.785

We evaluate the quality of the output (L, R) returned by each algorithm with respect to its β-value, the Adjusted Rand Index (ARI) (Gates & Ahn, 2017), as well as the ratio of the misclassified vertices defined by $\frac{|L \triangle C_1| + |R \triangle C_2|}{|L \cup C_1| + |R \cup C_2|}$, where $A \triangle B$ is the symmetric difference between A and B. All our reported results are the average performance of each algorithm over 10 runs, in which a random vertex from $C_1 \cup C_2$ is chosen as the starting vertex of the algorithm.

We first compare the algorithms' performance with different values of n_1, p_1 and q_1. As shown in Table 1, our algorithm not only runs faster, but also produces better clusters with respect to all three metrics. Secondly, since the clustering task becomes more challenging when the target clusters have higher β-value, we compare the algorithms' performance on a sequence of instances with increasing value of β. Since $q_1/p_1 = 2(1-\beta)/\beta$, we simply fix the values of n_1, p_1 as $n_1 = 1{,}000, p_1 = 0.001$, and generate graphs with increasing value of q_1/p_1; this gives us graphs with monotone values of β. As shown in Figure 5(a), our algorithm's performance is always better than the previous state-of-the-art.



(a) (b)

Figure 5. (a) The ARI of each algorithm when varying the target β-value. A larger q_1/p_1 ratio corresponds to a smaller β-value. (b) The ARI of each algorithm when bounding the runtime of the algorithm.

Thirdly, notice that both algorithms use some parameters to control the algorithm's runtime and the output's approximation ratio, which are naturally influenced by each other. To study this dependency, we generate graphs according to $n_1 = 100{,}000$, $p_1 = 0.000015$, and $q_1 = 0.00027$ which results in target sets with $\beta \approx 0.1$ and volume $\gamma \approx 6{,}000{,}000$. Figure 5(b) shows that, in comparison with the previous state-of-the-art, our algorithm takes much less time to produce output with the same ARI value.

Real-world Dataset. We demonstrate the significance of our algorithm on the Dyadic Militarised Interstate Disputes Dataset (v3.1) (Maoz et al., 2019), which records every interstate dispute during 1816–2010, including the level of hostility resulting from the dispute and the number of casualties, and has been widely studied in the social and political sciences (Mansfield et al., 2002; Martin et al., 2008) as well as the machine learning community (Hu et al., 2017; Menon & Elkan, 2011; Traag & Bruggeman, 2009). For a given time period, we construct a graph from the data by representing each country with a vertex and adding an edge between each pair of countries weighted according to the severity of any military disputes between those countries. Specifically, if there's a war[2] between the two countries, the corresponding two vertices are connected by an edge with weight 30; for any other dispute that is not part of an interstate war, the two corresponding vertices are connected by an edge with weight 1. We always use the USA as the starting vertex of the algorithm, and our algorithm's output, as visualised in Figure 1(a)-(d), can be well explained by geopolitics. The β-values of the pairs of clusters in Figures 1(a)-(d) are 0.361, 0.356, 0.170 and 0.191 respectively.

5.2. Results for Digraphs

Next we evaluate the performance of our algorithm for digraphs on synthetic and real-world datasets. Since there are no previous local digraph clustering algorithms that achieve similar objectives to ours, we compare the output of Algorithm 4 (`ECD`) with the state-of-the-art non-local algorithm proposed by Cucuringu et al. (2020), and we refer this to as `CLSZ` in the following.

[2]A war is defined by the maintainers of the dataset as a series of battles resulting in at least 1,000 deaths.

Table 3. Comparison of `EvoCutDirected` with `CLSZ` on the US migration dataset.

Figure 1 Subfigure	Algorithm	Cluster	Cut Imbalance	Flow Ratio
-	CLSZ	Pair 1	0.41	0.80
-	CLSZ	Pair 2	0.35	0.83
-	CLSZ	Pair 3	0.32	0.84
-	CLSZ	Pair 4	0.29	0.84
(e)	EvoCutDirected	Ohio Seed	0.50	0.56
(f)	EvoCutDirected	New York Seed	0.49	0.58
(g)	EvoCutDirected	California Seed	0.49	0.67
(h)	EvoCutDirected	Florida Seed	0.42	0.79

Synthetic Dataset. We first look at the *cyclic block model* (CBM) described in Cucuringu et al. (2020) with parameters k, n, p, q, and η. In this model, we generate a digraph with k clusters $C_1, \ldots, C_k$ of size n, and for $u, v \in C_i$, there is an edge between u and v with probability p and the edge direction is chosen uniformly at random. For $u \in C_i$ and $v \in C_{i+1 \mod k}$, there is an edge between u and v with probability q, and the edge is from u to v with probability η and from v to u with probability $1 - \eta$. We fix $p = 0.001$, $q = 0.01$, and $\eta = 0.9$.

Secondly, since the goal of our algorithm is to find local structure in a graph, we extend the cyclic block model with additional local clusters and refer to this model as CBM+. In addition to the parameters of the CBM, we introduce the parameters n', q'_1, q'_2, and η'. In this model, the clusters C_1 to C_k are generated as in the CBM, and there are two additional clusters C_{k+1} and C_{k+2} of size n'. For $u, v \in C_{k+i}$ for $i \in \{1, 2\}$, there is an edge between u and v with probability p and for $u \in C_{k+1}$ and $v \in C_{k+2}$, there is an edge with probability q'_1; the edge directions are chosen uniformly at random. For $u \in C_{k+1} \cup C_{k+2}$ and $v \in C_1$, there is an edge with probability q'_2. If $u \in C_{k+1}$, the orientation is from v to u with probability η' and from u to v with probability $1 - \eta'$; if $u \in C_{k+2}$, the orientation is from u to v with probability η' and from v to u with probability $1 - \eta'$. We always fix $q'_1 = 0.5$, $q'_2 = 0.005$, and $\eta' = 1$. Notice that the clusters C_{k+1} and C_{k+2} form a "local" cycle with the cluster C_1.

In Table 2, we report the average performance over 10 runs with a variety of parameters. We find that `CLSZ` can uncover the global structure in the CBM more accurately than `ECD`. On the other hand, `CLSZ` fails to identify the local cycle in the CBM+ model, while `ECD` succeeds.

Table 2. Comparison of `ECD` with `CLSZ` on synthetic data.

				Time		ARI	
Model	n'	n	k	ECD	CLSZ	ECD	CLSZ
CBM	-	10^3	5	**1.59**	3.99	0.92	**1.00**
CBM	-	10^3	50	**3.81**	156.24	0.99	0.99
CBM+	10^2	10^3	3	**0.24**	6.12	**0.98**	0.35
CBM+	10^2	10^4	3	**0.32**	45.17	**0.99**	0.01

Real-world Dataset. Finally, we evaluate the algorithms' performance on the US Migration Dataset (U.S. Census Bureau, 2000). For fair comparison, we follow Cucuringu et al. (2020) and construct the digraph as follows: every county in the mainland USA is represented by a vertex; for any vertices u, v, the edge weight of (u, v) is given by $\left|\frac{M_{u,v} - M_{v,u}}{M_{u,v} + M_{v,u}}\right|$, where $M_{u,v}$ is the number of people who migrated from county u to county v between 1995 and 2000; in addition, the direction of (u, v) is set to be from u to v if $M_{u,v} > M_{v,u}$, otherwise the direction is set to be the opposite.

For `CLSZ`, we follow their suggestion on the same dataset and set $k = 10$. Both algorithms' performance is evaluated with respect to the flow ratio, as well as the Cut Imbalance ratio used in their work. For any vertex sets L and R, the cut imbalance ratio is defined by $\mathrm{CI}(L, R) = \frac{1}{2} \cdot \left|\frac{e(L,R) - e(R,L)}{e(L,R) + e(R,L)}\right|$, and a higher $\mathrm{CI}(L, R)$ value indicates the connection between L and R is more significant. Using counties in Ohio, New York, California, and Florida as the starting vertices, our algorithm's outputs are visualised in Figures 1(e)-(h), and we compare them to the top 4 pairs returned by `CLSZ` in Table 3. Our algorithm produces better outputs with respect to both metrics.

These experiments suggest that local algorithms are not only more efficient, but also much more effective than non-local algorithms when learning certain structures in graphs. In particular, some localised structure might be hidden when applying the objective function over the entire graph.

Acknowledgements

We thank Luca Zanetti for making insightful comments on an earlier version of this work. Peter Macgregor is supported by the Langmuir PhD Scholarship, and He Sun is supported by an EPSRC Early Career Fellowship (EP/T00729X/1).

References

Andersen, R. A local algorithm for finding dense subgraphs. *ACM Transactions on Algorithms (TALG)*, 6 (4):1–12, 2010.

Andersen, R. and Chellapilla, K. Finding dense subgraphs with size bounds. In *International Workshop on Algorithms and Models for the Web-Graph*, pp. 25–37, 2009.

Andersen, R. and Chung, F. Detecting sharp drops in Pagerank and a simplified local partitioning algorithm. In *International Conference on Theory and Applications of Models of Computation (TAMC'07)*, pp. 1–12, 2007.

Andersen, R., Chung, F., and Lang, K. Local graph partitioning using Pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 475–486, 2006.

Andersen, R., Gharan, S. O., Peres, Y., and Trevisan, L. Almost optimal local graph clustering using evolving sets. *Journal of the ACM (JACM)*, 63(2):1–31, 2016.

Benson, A. R., Gleich, D. F., and Leskovec, J. Tensor spectral clustering for partitioning higher-order network structures. In *15th International Conference on Data Mining (ICDM'15)*, pp. 118–126, 2015.

Benson, A. R., Gleich, D. F., and Leskovec, J. Higher-order organization of complex networks. *Science*, 353(6295): 163–166, 2016.

Cohen, M. B., Kelner, J., Peebles, J., Peng, R., Rao, A. B., Sidford, A., and Vladu, A. Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *49th Annual ACM Symposium on Theory of Computing (STOC'17)*, pp. 410–419, 2017.

Cucuringu, M., Li, H., Sun, H., and Zanetti, L. Hermitian matrices for clustering directed graphs: Insights and applications. In *23rd International Conference on Artificial Intelligence and Statistics (AISTATS'20)*, pp. 983–992, 2020.

Fountoulakis, K., Wang, D., and Yang, S. p-norm flow diffusion for local graph clustering. In *37th International Conference on Machine Learning (ICML'20)*, pp. 3222–3232, 2020.

Gates, A. J. and Ahn, Y.-Y. The impact of random models on clustering similarity. *The Journal of Machine Learning Research*, 18(1):3049–3076, 2017.

Hu, C., Rai, P., and Carin, L. Deep generative models for relational data with side information. In *34th International Conference on Machine Learning (ICML'17)*, pp. 1578–1586, 2017.

Laenen, S. and Sun, H. Higher-order spectral clustering of directed graphs. In *34th Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020.

Li, A. and Peng, P. Detecting and characterizing small dense bipartite-like subgraphs by the bipartiteness ratio measure. In *24th International Symposium on Algorithms and Computation (ISAAC'13)*, pp. 655–665, 2013.

Liu, M. and Gleich, D. F. Strongly local p-norm-cut algorithms for semi-supervised learning and local graph clustering. In *34th Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020.

Lovász, L. and Simonovits, M. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *31st Annual IEEE Symposium on Foundations of Computer Science (FOCS'90)*, pp. 346–354, 1990.

Mansfield, E. D., Milner, H. V., and Rosendorff, B. P. Why democracies cooperate more: Electoral control and international trade agreements. *International Organization*, 56(3):477–513, 2002.

Maoz, Z., Johnson, P. L., Kaplan, J., Ogunkoya, F., and Shreve, A. P. The dyadic militarized interstate disputes (MIDs) dataset version 3.0: Logic, characteristics, and comparisons to alternative datasets. *Journal of Conflict Resolution*, 63(3):811–835, 2019. URL `https://correlatesofwar.org/`. Accessed: January 2021.

Martin, P., Mayer, T., and Thoenig, M. Make trade not war? *The Review of Economic Studies*, 75(3):865–900, 2008.

Menon, A. K. and Elkan, C. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pp. 437–452, 2011.

Moore, C., Yan, X., Zhu, Y., Rouquier, J.-B., and Lane, T. Active learning for node classification in assortative and disassortative networks. In *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pp. 841–849, 2011.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-GCN: Geometric Graph Convolutional Networks. In *7th International Conference on Learning Representations (ICLR'19)*, 2019.

Takai, Y., Miyauchi, A., Ikeda, M., and Yoshida, Y. Hypergraph clustering based on pagerank. In *26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'20)*, pp. 1970–1978, 2020.

Traag, V. A. and Bruggeman, J. Community detection in networks with positive and negative links. *Physical Review E*, 80(3):036115, 2009.

Trevisan, L. Max cut and the smallest eigenvalue. *SIAM Journal on Computing (SICOMP)*, 41(6):1769–1786, 2012.

U.S. Census Bureau. United states 2000 census. https://web.archive.org/web/20150905081016/https://www.census.gov/population/www/cen2000/commuting/, 2000. Accessed: January 2021.

Wang, D., Fountoulakis, K., Henzinger, M., Mahoney, M. W., and Rao, S. Capacity releasing diffusion for speed and locality. In *34th International Conference on Machine Learning (ICML'17)*, pp. 3598–3607, 2017.

Yin, H., Benson, A. R., Leskovec, J., and Gleich, D. F. Local higher-order graph clustering. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17)*, pp. 555–564, 2017.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. In *34th Advances in Neural Information Processing Systems (NeurIPS'20)*, 2020.

Zhu, Z. A., Lattanzi, S., and Mirrokni, V. A local algorithm for finding well-connected clusters. In *30th International Conference on Machine Learning (ICML'13)*, pp. 396–404, 2013.