Capacity Releasing Diffusion for Speed and Locality

Di Wang [1] Kimon Fountoulakis [2] Monika Henzinger [3] Michael W. Mahoney [2] Satish Rao [1]

Abstract

Diffusions and related random walk procedures are of central importance in many areas of machine learning, data analysis, and applied mathematics. Because they spread mass agnostically at each step in an iterative manner, they can sometimes spread mass "too aggressively," thereby failing to find the "right" clusters. We introduce a novel *Capacity Releasing Diffusion (CRD) Process*, which is both faster and stays more local than the classical spectral diffusion process. As an application, we use our CRD Process to develop an improved local algorithm for graph clustering. Our local graph clustering method can find local clusters in a model of clustering where one begins the CRD Process in a cluster whose vertices are connected better internally than externally by an $O(\log^2 n)$ factor, where n is the number of nodes in the cluster. Thus, our CRD Process is the first local graph clustering algorithm that is not subject to the well-known quadratic Cheeger barrier. Our result requires a certain smoothness condition, which we expect to be an artifact of our analysis. Our empirical evaluation demonstrates improved results, in particular for realistic social graphs where there are moderately good—but not very good—clusters.

1. Introduction

Diffusions and related random walk procedures are of central importance in many areas of machine learning, data analysis, and applied mathematics, perhaps most conspicuously in the area of spectral clustering (Cheeger, 1969; Donath & Hoffman, 1973; von Luxburg, 2006; Shi & Malik, 2000), community detection in networks (Ng et al., 2001; White & Smyth, 2005; Leskovec et al., 2009; Jeub et al., 2015), so-called manifold learning (Belkin & Niyogi, 2003; Mahoney et al., 2012), and PageRank-based spectral ranking in web ranking (Page et al., 1999; Gleich, 2015). Particularly relevant for our results are local/personalized versions of PageRank (Jeh & Widom, 2003) and local/distributed versions of spectral clustering (Spielman & Teng, 2004; Andersen et al., 2006; Andersen & Peres, 2009). These latter algorithms can be used to find provably-good small-sized clusters in very large graphs without even touching the entire graph; they have been implemented and applied to billion-node graphs (Shun et al., 2016); and they have been used to characterize the clustering and community structure in a wide range of social and information networks (Leskovec et al., 2009; Jeub et al., 2015).

Somewhat more formally, we will use the term *diffusion* on a graph to refer to a process that spreads mass among vertices by sending mass along edges step by step according to some rule. With this interpretation, classical *spectral diffusion* spreads mass by distributing the mass on a given node equally to the neighbors of that node in an iterative manner. A well-known problem with spectral methods is that—due to their close relationship with random walks—they sometimes spread mass "too aggressively," and thereby they don't find the "right" partition. In theory, this can be seen with so-called Cockroach Graph (Guattery & Miller, 1998; von Luxburg, 2006). In practice, this is seen by the extreme sensitivity of spectral methods to high-degree nodes and other structural heterogeneities in real-world graphs constructed from very noisy data (Leskovec et al., 2009; Jeub et al., 2015). More generally, it is well-known that spectral methods can be very sensitive to a small number of random edges, e.g., in small-world graphs, that "short circuit" very distant parts of the original graph, as well as other noise properties in realistic data. Empirically, this is well-known to be a particular problem when there are moderately good—but not very good—clusters in the data, a situation that is all too common in machine learning and data analysis applications (Jeub et al., 2015).

Here, we introduce a novel *Capacity Releasing Diffusion (CRD) Process* to address this problem. Our CRD Process is a type of diffusion that spreads mass according to a carefully-constructed push-relabel rule, using techniques that are well-known from flow-based graph algorithms, but modified here to release the capacity of edges to transmit mass. Our CRD Process has better properties with respect to limiting the spread of mass inside local well-connected

[1]EECS, UC Berkeley, Berkeley, CA, USA [2]ICSI and Statistics, UC Berkeley, Berkeley, CA, USA [3]Computer Science, University of Vienna, Vienna, Austria. Correspondence to: Di Wang <wangd@eecs.berkeley.edu>.

clusters. It does so with improved running time properties. We show that this yields improved local clustering algorithms, both in worst-case theory and in empirical practice.

1.1. Capacity Releasing Diffusion (CRD)

We start by describing the *generic* CRD process in Figures 1 and 2, which lays down the dynamics of spreading mass across the graph. Importantly, this dynamical process is independent of any particular task to which it may be applied. Later (in Section 2) we also present a concrete CRD algorithm for the specific task of local clustering that exploits the dynamics of the generic CRD process[1].

1. Begin with $2d(u)$ mass at a single (given) vertex u.
2. Repeatedly perform a CRD step, and then double the mass at every vertex.

Figure 1. **Generic Capacity Releasing Diffusion (CRD) Process**

The entire CRD process (Figure 1) repeatedly applies the generic CRD inner process (which we call a *CRD step*), and then it doubles the amount of mass at all vertices between invocations. A CRD step starts with each vertex u having mass $m(u) \leq 2d(u)$, where $d(u)$ is the degree of u, and spreads the mass so that at the end each vertex u has mass $m(u) \leq d(u)$. Observe that, essentially, each CRD step spreads the mass to a region of roughly twice the volume comparing to the previous step.

Each vertex v initially has mass $m(v) \leq 2d(v)$ and needs to spread the mass so that $m(v) \leq d(v) \forall v$.

1. Each vertex v maintains a label, $l(v)$, initially set to 0.
2. Each edge e maintains $m(e)$, which is the mass moved from u to v ,where $m(v,u) = -m(u,v)$. We note that both $l(v)$ and $m(u,v)$ are variables local to this inner process, where $m(v)$ evolves across calls to this process.
3. An edge, $e = (u,v)$, is *eligible* with respect to the labeling if it is *downhill*: $l(u) > l(v)$, and the mass $m(u,v)$ moved along (u,v) is less than $l(u)$.
4. The *excess* of a vertex is $\mathrm{ex}(v) \overset{\text{def}}{=} \max(0, m(v) - d(v))$.

The inner process continues as long as there is a vertex v with $\mathrm{ex}(v) > 0$; it either sends mass over an eligible edge incident to v, or if there is none, then it increases $l(v)$ by 1.

Figure 2. **Generic CRD Inner Process.**

[1]The relation between the generic CRD process and the CRD algorithm for local graph clustering is analogous to the relation between local random walks and a local spectral graph partitioning algorithm such as that of Andersen et al. (2006).

The generic CRD inner process (Figure 2) implements a modification of the classic "push-relabel" algorithm (Goldberg & Tarjan, 1988; 2014) for routing a source-sink flow. The crucial property of our process (different from the standard push-relabel) is that edge capacity is made available to the process slowly by *releasing*. That is, we only allow $l(u)$ units of mass to move across any edge (u,v), where $l(u)$ is the label (or height) maintained by the CRD inner process. Thus, edge capacity is *released* to allow mass to cross the edge as the label of the endpoint rises. As we will see, this difference is critical to the theoretical and empirical performance of the CRD algorithm.

1.2. Example: Classical Versus Capacity Releasing

To give insight into the differences between classical spectral diffusion and our CRD Process, consider the graph in Figure 3. There is a "cluster" B, which consists of k paths, each of length l, joined at a common node u. There is one edge from u to the rest of the graph, and we assume the other endpoint v has very high degree such that the vast majority of the mass arriving there is absorbed by its neighbors in $\overline{B}$. While idealized, such an example is not completely unrealistic (Leskovec et al., 2009; Jeub et al., 2015).

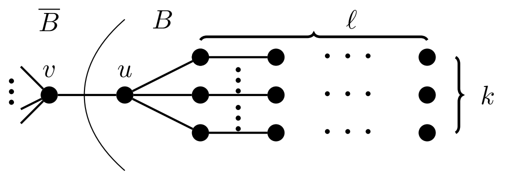

Figure 3. An example where Capacity Releasing Diffusion beats classical spectral diffusion by $\Omega(\ell)$.

Consider first classical spectral diffusion, with a random walk starting from some vertex in B. This process requires $\Omega(\ell^2)$ steps to spread probability mass to a constant fraction of the nodes on the paths, and in this many steps, the expected number of times to visit u is $\Omega(\ell)$. Because of the edge to v, each time we visit u, we have a $\Omega(1/k)$ chance of leaving B. Thus, when ℓ is $\Omega(k)$, the random walk is expected to leave B and never return, i.e., the classical diffusion will leak out all the probability mass before even spreading beyond a constant fraction of B.

Consider next our CRD Process, starting with mass at the vertex $u \in B$ (which would be a worst-case starting node in B for CRD). Assume that at some point the mass is spread along z neighboring vertices on each of the k paths. To continue the spread to $2z$ vertices in the next CRD step, the labels will be raised to (at most) $2z$ to allow the mass to spread over the path of length $2z$. This enables the spread along the paths, but it only releases a capacity of $2z$ to the exiting edge (u,v). Since in this call, a total of $2zk$ mass is in the set B, at most $1/k$ of the mass escapes. After

$\log \ell$ CRD steps, the mass is spread over all the k length-ℓ paths, and only a $(2\log \ell)/k$ fraction of the mass has escaped from B. Thus if $\ell = \Omega(k)$, as before, a factor of $k/\log \ell$ less mass has escaped from B with the CRD Process than with the classical diffusion process.

Without the releasing, however, the mass escaping B would be large, as even raising the label of vertex u to 1 would allow an arbitrary amount of mass to leak out.

Finally, note that the ℓ^2 mixing time makes spectral diffusions a $\Omega(\ell)$ factor slower than CRD. This drawback of spectral techniques can perhaps be resolved using sophisticated methods such as evolving sets (Andersen & Peres, 2009), though it comes easily with CRD.

1.3. Our Main Results

We provide theoretical and empirical evidence that our CRD algorithm is superior to classical diffusion methods at finding clusters, with respect to noise tolerance, recovery accuracy, cut conductance, and running time. Here the *cut conductance* $\Phi(S)$ of a cut $(S, \bar{S})$ is $\Phi(S) := \frac{|E(S,\bar{S})|}{\min(\text{vol}(S),\text{vol}(V\setminus S))}$, where $E(S, \bar{S})$ denotes the set of edges between S and $\bar{S}$,[2] and the *volume* $\text{vol}(S)$ is the sum of the degrees of the vertices in S. In all these measures, we break the quadratic Cheeger barrier for classical diffusions (explained below) while presenting a *local* algorithm (i.e., an algorithm whose running time depends on the volume of the cluster found and *not* of the whole graph).

Our first main result (Section 2) presents a CRD algorithm and its running time. The CRD algorithm is a parameterized specialization of the generic CRD Process, where we limit the maximum label of vertices, as well as the maximum edge capacity. We prove that this specialization is efficient, in that it runs in time linear in the total mass and the label limit, and it either succeeds in spreading the mass or it leaves all unspread mass at nodes with high label. This property is analogous to the ispoerimetric capacity control provided by local spectral methods, and it is important for locating cluster bottlenecks. We use this crucially in our context to find low conductance clusters.

Our second main result (Section 3) concerns the use of the CRD algorithm to find good local clusters in large graphs. Our result posits the existence of a "good" cluster B, which satisfies certain conditions (Assumption 1 and 2) that naturally capture the notion of a local structure. The rather weak Assumption 1 states that B's internal connectivity $\phi_S(B)$ (see Section 3 for definition) is a constant factor better (i.e., larger) than the conductance $\phi(B)$. Assumption 2 states that we have a smoothness condition which needs that any subset $T \subset B$ has $\text{polylog}(\text{vol}(B))$ times more neighbors in $B - T$ than in $V - B$. Under these conditions, we can recover B starting from any vertex in B.

Both assumptions formalize the idea that the signal of the local structure is stronger than the noise of the cluster by some moderately large factor. More specifically, Assumption 1 roughly says that the weakest signal of any subset of B is a constant times stronger than the average noise of B; and Assumption 2 roughly says the signal of any subset is $\text{polylog}(\text{vol}(B))$ times stronger than the noise of the subset.

We note that Assumption 1 is significantly weaker than the factor in Zhu et al. (2013), where it is shown how to localize a cluster B such that $\phi_S(B) \geq \sqrt{\phi(B)}$. Their condition is considerably stricter than our condition on the ratio between $\phi_S(B)$ and $\phi(B)$, especially when $\phi(B)$ is small, as is common. Their algorithm relies on proving that a classical diffusion starting at a typical node keeps most of its mass inside of B. However, they do not need something like our smoothness condition.

With the additional smoothness condition, we break the dependence on $\sqrt{\phi(B)}$ that is central to all approaches using spectral diffusions, including Zhu et al. (2013), for the first time with a local algorithm. In particular, comparing to Zhu et al. (2013), under their parameter settings (but with the smoothness condition), we identify a cluster with $\sqrt{\phi(B)}$ times less error, and we have a $1/\sqrt{\phi(B)}$ speedup in running time. This improvement is (up to a $\log \ell$-factor) consistent with the behavior in the example of the previous section where the improvement is $k/\log \ell = O(1/(\sqrt{1/\phi(B)} \log \ell))$ as $\phi(B) = 1/k\ell$ and $\ell = \Omega(k)$.

We note that with the additional smoothness condition, our theoretical results hold for any starting node v_s in B, in contrast to prior spectral-based results which only work when starting from a "good" node (where only a constant fraction of the nodes in B are good). We expect the smoothness condition to be an artifact of our analysis, i.e., similar results actually hold when starting at good nodes in B, even without this assumption.

Our third main result (Section 4) is an empirical illustration of our method. We consider several social and information networks studied previously that are particularly challenging for spectral methods. In particular, while graphs that have upward-sloping NCPs (Network Community Profiles) have good small clusters (Leskovec et al., 2009; Jeub et al., 2015), denser social networks with flat NCPs do not have any *very*-good conductance clusters of any size. They do, however, often have *moderately*-good clusters, but these are very difficult for spectral methods to identify (Jeub et al., 2015). Our empirical results show that our CRD-based local clustering algorithm is better able to identify and extract in a strongly local running time moderately

[2]Unless otherwise noted, when speaking of the conductance of a cut S, we assume S to be the side of minimum volume.

good quality clusters from several such social networks.

1.4. Previous Work: Low Conductance Cuts, Diffusions, and Multicommodity Flow

Spectral algorithms for computing eigenvalues use some variant of repeated matrix multiplication, which for graphs is a type of classical diffusion. For the Laplacian of a graph, the convergence rate is $O(1/\lambda_2)$, where λ_2 is the second smallest eigenvalue by of this matrix. The Lanczos method improves this rate to $O(\sqrt{1/\lambda_2})$ by cleverly and efficiently combining different iterations of the diffusions. See, e.g., Orecchia et al. (2012) for more details on this.

One application of such a computation is to find a low conductance cut in a graph. The second eigenvector for G can be used to find a cut of conductance $O(\frac{1}{\lambda_2})$ (Cheeger, 1969; Donath & Hoffman, 1973). Let ϕ_G be the minimum conductance in the graph. In his work, Cheeger already observed that random-walk based diffusion can make a $\Theta(1/\sqrt{\phi_G})$ error in estimating the conductance, informally known as the (quadratic) *Cheeger barrier*, and illustrated in our example. This, combined with the fact that $\lambda_2 = O(1/\phi_G)$, gives a spectral method to find an $O(\phi_G^{1/2})$ conductance cut in G.

Spielman-Teng (2004) used local versions of diffusions (i.e., those with small support) to compute recursive decompositions efficiently, and then they used locality to produce linear time partitioning algorithms. Andersen, Chung and Lang (2006) developed an improved version that adjusts the standard diffusion by having mass settled at vertices, resulting in significantly improved bounds to $O(\sqrt{\phi_G \log n})$ on the conductance of the returned cut $(B, \bar{B})$ in time $\tilde{O}(\frac{\text{vol}(B)}{\phi_G})$. Allen-Zhu, Lattanzi and Mirrokni (2013) analyzed the behavior of the same algorithm under certain well-connected conditions. The EvoCut algorithm of Andersen and Peres (2009) improved the running time of this method to $\tilde{O}(\frac{\text{vol}(B)}{\sqrt{\phi_G}})$. As all these methods are based on spectral diffusion, their performance with respect to conductance is subject to the Cheeger barrier. Other processes have been proposed for random walks that mix faster, e.g., non-backtracking random walks (Alon et al., 2007). These too are subject to the Cheeger barrier asymptotically. Our result is the first to break this barrier in any broad setting, where classical spectral methods fail.

Multicommodity flow based methods are able to find clusters of conductance $O(\phi_G \log n)$ (Leighton & Rao, 1988), bypassing the limit inherent in purely spectral methods. A semidefinite programming approach, which can be viewed as combining multicommodity flow and spectral methods, yields cuts of conductance $O(\phi_G\sqrt{\log n})$ (Arora et al., 2009). These algorithms are very non-local, e.g., in the sense that their running time depends on the size of the whole graph, and it is not clear that they can be meaningfully localized. We do, however, use well-known flow-based ideas in our algorithm. In particular, recall that push-relabel and in general "shortest-path" based methods have a celebrated history in algorithms (Goldberg & Tarjan, 2014). Using levels to release capacity, however, as we do in our algorithm, is (to our knowledge) completely new.

2. Capacity Releasing Diffusion

In this section, we describe our algorithm which implements a specific version of the generic CRD Process. In particular, it has some modifications for efficiency reasons, and it terminates the diffusion when it finds a bottleneck during the process. The algorithm iteratively calls a subroutine *CRD-inner*, which implements one CRD step.

For efficiency reasons, *CRD-inner* doesn't necessarily carry out a full CRD step, where a *full CRD step* means every node u has at most $d(u)$ mass at termination. In particular, *CRD-inner* only makes a certain amount of "effort" (which is tuned by a parameter ϕ) to spread the mass, and if there is a bottleneck in the form of a cut that requires "too much effort" for the diffusion to get through, then *CRD-inner* may leave excess mass on nodes, i.e., $m(v) > d(v)$ at termination. More specifically, given ϕ, *CRD-inner* guarantees to overcome any bottleneck of conductance $\Omega(\phi)$, i.e., if it doesn't carry out a full CRD step, then it returns a cut of conductance $O(\phi)$ as a certificate. We will discuss *CRD-inner* with more detail in Section 2.2.

2.1. CRD Algorithm

Given a starting node v_s, the CRD algorithm (Algorithm 1) is essentially the CRD Process starting from v_s, as described in Figure 1. The algorithm takes as input a parameter ϕ, which is used to tune *CRD-inner*. Since *CRD-inner* may stop short of a full CRD step due to a bottleneck, we remove any excess mass remaining on nodes after calling *CRD-inner*. Due to the excess removal, we may discard mass as the algorithm proceeds. In particular, as we start with $2d(v_s)$ mass, and double the amount after every CRD step, the amount of mass after the j-th doubling is $2d(v_s) \cdot 2^j$ if we never remove excess. When the actual amount of mass is significantly smaller than $2d(v_s) \cdot 2^j$, there must be a bottleneck $(K, \bar{K})$ during the last CRD step, such that K contains a large fraction of the mass (and of the excess) and such that *CRD-inner* cannot push any more mass from K to $\bar{K}$. We terminate the CRD algorithm when this happens, as the mass and, as we can show, thus the volume of K must be large, while there are few edges between K and $\bar{K}$. Thus K is a low-conductance cluster around v_s. Formally, the algorithm takes input parameters τ and t, and it terminates either when the amount of mass drops below $\tau(2d(v_s) \cdot 2^j)$ after iteration j, or after iteration t if the former never happens. It returns the mass on the nodes (i.e., $m(\cdot)$), as well as the cut K returned by the

Algorithm 1 *CRD Algorithm*(G, v_s, ϕ, τ, t)

```
. Initialization:
  $m(v_s) = d(v_s), m(v) = 0, \forall v \neq v_s; j = 0.$
. For $j = 0, \ldots, t$
. . $m(v) \leftarrow 2m(v), \forall v$
. . Assertion: $m(v) \leq 2d(v), \forall v$
. . Call CRD-inner with $G, m(\cdot), \phi$, get cut $K_j$
    ($K_j$ empty if CRD-inner finishes full CRD step).
. . $m(v) \leftarrow \min(m(v), d(v)), \forall v$
. . If $\sum_v m(v) \leq \tau(2d(v_s) \cdot 2^j)$
. . . Return $m(\cdot)$, and $K \overset{\text{def}}{=} K_j$. Terminate.
. End For
. Return $m(\cdot), K \overset{\text{def}}{=} K_t$.
```

last *CRD-inner* call in the former termination state.

The running time of our CRD algorithm is local (i.e., proportional to the volume of the region it spreads mass to, rather than the volume of the entire graph). In particular, each *CRD-inner* call takes time linear in the amount of mass, and as the amount of mass increases geometrically before we terminate, the running time of the CRD algorithm is dominated by the last *CRD-inner* call.

2.2. CRD Inner Procedure

Now we discuss the *CRD-inner* subroutine (Algorithm 2), which aims to carry out one CRD step. In particular, each node v has $m(v) \leq 2d(v)$ mass at the beginning, and *CRD-inner* tries to spread the mass so each node v has $m(v) \leq d(v)$ mass at the end. Not surprisingly, as the CRD step draws intuition from flow routing, our *CRD-inner* can be viewed as a modification of the classic push-relabel algorithm.

As described in Figure 2, we maintain a label $l(v)$ for each node v, and the net mass being pushed along each edge. Although the graph is undirected, we consider each edge $e = \{u, v\}$ as two directed arcs (u, v) and (v, u), and we use $m(u, v)$ to denote the net mass pushed from u to v (during the current *CRD-inner* invocation). Under this notation, we have $m(u, v) = -m(v, u)$. We denote $|m(\cdot)| \overset{\text{def}}{=} \sum_v m(v)$ as the total amount of mass, $\text{ex}(v) \overset{\text{def}}{=} \max(m(v) - d(v), 0)$ as the amount of excess on v, and we let ϕ be the input parameter tuning the "effort" made by *CRD-inner* (which will be clear shortly).

As noted earlier, to make *CRD-inner* efficient, we deviate from the generic CRD step. In particular, we make the following modifications:

1. The label of any node can be at most $h = 3 \log |m(\cdot)|/\phi$. If v is raised to level h, but still has excess mass, *CRD-inner* leaves the excess on v, and won't work on v any more. Formally, v is *active* if $l(v) < h$ and $\text{ex}(v) > 0$. We keep a list Q of all active nodes, and terminate *CRD-inner* when Q is empty.
2. In addition to capacity releasing, the net mass along any edge can be at most $C = 1/\phi$. Formally, for an arc (v, u), its *effective capacity* is $\hat{c}(v, u) \overset{\text{def}}{=} \min(l(v), C)$, and its *residual capacity* is $r_m(v, u) \overset{\text{def}}{=} \hat{c}(v, u) - m(v, u)$. The arc (v, u) is *eligible* iff $l(v) > l(u)$ (i.e., downhill) and $r_m(v, u) > 0$. We only push mass along eligible arcs.
3. We enforce $m(v) \leq 2d(v)$ for all v through the execution. This is assumed at the start, and we never push mass to v if that would result in $m(v) > 2d(v)$.

The parameter ϕ in the first two modifications limits the work done by *CRD-inner*, and it captures how hard *CRD-inner* will try to carry out the full CRD step (e.g., when h, C are infinitely large, *CRD-inner* implements the full CRD step). Given any ϕ, *CRD-inner* makes enough effort by allowing nodes to have height up to h and by using the above edge capacities to overcome bottlenecks of conductance $\Omega(\phi)$ during the diffusion process. If it doesn't finish the full CRD step, then it returns a cut of conductance $O(\phi)$ as certificate.

Another motivation of tuning with parameter ϕ is to keep the diffusion local. Since *CRD-inner* doesn't try to get through low-conductance bottlenecks, the diffusion tends to spread mass over well-connected region, instead of pushing mass out of a bottleneck. This guarantees that the work performed is linear in the volume of the returned cluster, i.e., that it is a strongly local algorithm, since only a small fraction of mass can leak out of the cluster.

The third modification guarantees when *CRD-inner* terminates with a lot of excess on nodes, the excess won't be concentrated on a few nodes, as no node can have more mass than twice its degree, and thus the cut returned must contain a large region.

We have the following theorem for *CRD-inner*.

Theorem 1. *Given* $G, m(\cdot)$*, and* $\phi \in (0, 1]$*, such that* $|m(\cdot)| \leq \text{vol}(G)$*, and* $\forall v : m(v) \leq 2d(v)$ *at the start,* CRD-inner *terminates with one of the following cases:*

1. CRD-inner *finishes the full CRD step:* $\forall v : m(v) \leq d(v)$.
2. *There are nodes with excess, and we can find a cut* A *of conductance* $O(\phi)$*. Moreover,* $\forall v \in A : 2d(v) \geq m(v) \geq d(v)$*, and* $\forall v \in \bar{A} : m(v) \leq d(v)$.

The running time is $O(|m(\cdot)| \log(|m(\cdot)|)/\phi)$.

Proof sketch. Let $l(\cdot)$ be the labels of nodes at termination. First note all nodes with excess must be on level h. Moreover, since we only push from a node v if it has excess (i.e., $m(v) \geq d(v)$), once a node has at least $d(v)$

Algorithm 2 *CRD-inner*(G,$m(\cdot)$,ϕ)

```
. Initialization:
. . $\forall\{v,u\} \in E, m(u,v) = m(v,u) = 0; \forall v, l(v) = 0$
. . $Q = \{v|m(v) > d(v)\}, h = \frac{3\log|m(\cdot)|}{\phi}$
. While $Q$ is not empty
. . Let $v$ be the lowest labeled node in $Q$.
. . Push/Relabel($v$).
. . If Push/Relabel($v$) pushes mass along $(v,u)$
. . . If $v$ becomes in-active, remove $v$ from $Q$
. . . If $u$ becomes active, add $u$ to $Q$
. . Else If Push/Relabel($v$) increases $l(v)$ by 1
. . . If $l(v) = h$, remove $v$ from $Q$.
```

```
Push/Relabel($v$)
. If there is any eligible arc $(v,u)$
. . Push($v,u$).
. Else
. . Relabel($v$).
```

```
Push($v,u$)
. $\psi = \min(\mathrm{ex}(v), r_m(v,u), 2d(u) - m(u))$
. Push $\psi$ units of mass from $v$ to $u$:
  $m(v,u) \leftarrow m(v,u) + \psi, m(u,v) \leftarrow m(u,v) - \psi$;
  $m(v) \leftarrow m(v) - \psi, m(u) \leftarrow m(u) + \psi$.
```

```
Relabel($v$)
. $l(v) \leftarrow l(v) + 1$.
```

mass, it always has at least $d(v)$ mass. Note further that $l(v) \geq 1$ if and only if $\mathrm{ex}(v) > 0$ at some point during the process. Thus, we know the following: $l(v) = h \Rightarrow 2d(v) \geq m(v) \geq d(v)$; $h > l(v) \geq 1 \Rightarrow m(v) = d(v)$; $l(v) = 0 \Rightarrow m(v) \leq d(v)$.

Let $B_i = \{v|l(v) = i\}$. Since the total amount of mass $|m(\cdot)|$ is at most the volume of the graph, if $B_0 = \emptyset$ or $B_h = \emptyset$, then we have case (1) of the theorem.

Otherwise, both B_h and B_0 are non-empty. Let the *level cut* $S_i = \cup_{j=i}^{h} B_j$ be the set of nodes with label at least i. We have h level cuts $S_h, \ldots, S_1$, where $\mathrm{vol}(S_h) \geq 1$, and $S_j \subseteq S_i$ if $j > i$. The conductance of these cuts, when we go from S_h down to S_1, lower bounds how much the volume grows from S_h to S_1. If all these cuts have $\Omega(\phi)$ conductance, by our choice of h, the volume of S_1 will be much larger than $|m(\cdot)|$. This gives a contradiction, since any node $v \in S_1$ has $m(v) \geq d(v)$, and we don't have enough mass. It follows that at least one of the level cuts has conductance $O(\phi)$.

As to the running time, the graph G is given implicitly, and we only acess the list of edges of a node when it is active. Each active node v has $d(v)$ mass, and the total amount of mass is $|m(\cdot)|$, so the algorithm touches a region of volume at most $|m(\cdot)|$. Thus, the running time has linear dependence on $|m(\cdot)|$. Using an amortization argument one can show that the total work of the subroutine (in the worst case) is $O(|m(\cdot)|h) = O(|m(\cdot)|\log(|m(\cdot)|)/\phi)$. □

There are certain details in the implementation of *CRD-inner* that we don't fully specify, such as how to check if v has any outgoing eligible arcs, and how (and why) we pick the active node with lowest label. These details are important for the running time to be efficient, but don't change the dynamics of the diffusion process. Most of these details are standard to push-relabel framework, and we include them (as well as the detailed proof of Theorem **??**) in Appendix A.

3. Local Graph Clustering

In this section, we provide theoretical evidence that the CRD algorithm can identify a good local cluster in a large graph if there exists one around the starting node. We define *set conductance,* $\phi_S(B)$ (or *internal connectivity*) of a set $B \subset V$ is the minimum conductance of any cut in the induced subgraph on B.

Informally, for a "good" cluster B, any inside bottleneck should have larger conductance than $\phi(B)$, and nodes in B should be more connected to other nodes inside B than to nodes outside. We capture the intuition formally as follows.

Assumption 1. $\sigma_1 \overset{\text{def}}{=} \frac{\phi_S(B)}{\phi(B)} \geq \Omega(1)$.

Assumption 2. *There exists* $\sigma_2 \geq \Omega(1)$*, such that any* $T \subset B$ *with* $vol_B(T) \leq \mathrm{vol}_B(B)/2$ *satisfies*

$$\frac{|E(T, B \setminus T)|}{|E(T, V \setminus B)| \log \mathrm{vol}(B) \log \frac{1}{\phi_S(B)}} \geq \sigma_2.$$

Following prior work in local clustering, we formulate the goal as a promise problem, where we assume the existence of an unknown target good cluster $B \subset V$ satisfying Assumption 1 and 2. In the context of local clustering, we also assume $\mathrm{vol}(B) \leq \mathrm{vol}(G)/2$. Similar to prior work, we assume the knowledge of a node $v_s \in B$, and rough estimates (i.e., within constant factor) of the value of $\phi_S(B)$ and $\mathrm{vol}(B)$. We use the CRD algorithm with v_s as the starting node, $\phi = \Theta(\phi_S(B))$, $\tau = 0.5$, and $t = \Theta(\log \frac{\mathrm{vol}(B)}{d(v_s)})$. With the parameters we use, the algorithm will terminate due to too much excess removed, i.e., $|m(\cdot)| \leq \tau(2d(v_s) \cdot 2^j)$ after some iteration j. The region where the diffusion spreads enough mass will be a good approximation of B.

Theorem 2. *Starting from any* $v_s \in B$*, with the above parameters, when the CRD algorithm terminates, if we let* $S = \{v|m(v) \geq d(v)\}$*, then we have:*

1. $\mathrm{vol}(S \setminus B) \leq O(\frac{1}{\sigma}) \cdot \mathrm{vol}(B)$
2. $\mathrm{vol}(B \setminus S) \leq O(\frac{1}{\sigma}) \cdot \mathrm{vol}(B)$

where $\sigma = \min(\sigma_1, \sigma_2) \geq \Omega(1)$*, with the* σ_1, σ_2 *from Assumption 1 and 2. The running time is* $O(\frac{\mathrm{vol}(B)\log \mathrm{vol}(B)}{\phi_S(B)})$.

The theorem states that the cluster recovered by the CRD algorithm has both good (degree weighted) precision and recall with respect to B; and that the stronger the "signal" (relative to the "noise"), i.e., the larger σ_1, σ_2, the more accurate our result approximates B.

If the goal is to minimize conductance, then we can run one extra iteration of the CRD algorithm after termination with a smaller value for ϕ (not necessarily $\Theta(\phi_S(B))$ as used in previous iterations). In this case, we have the following.

Theorem 3. *If we run the CRD algorithm for one extra iteration, with $\phi \geq \Omega(\phi(B))$, then* CRD-inner *will end with case* (2) *of Theorem 1. Let K be the cut returned. We have:*

1. $\mathrm{vol}(K \setminus B) \leq O(\frac{\phi(B)}{\phi}) \cdot \mathrm{vol}(B)$
2. $\mathrm{vol}(B \setminus K) \leq O(\frac{\phi(B)}{\phi_S(B)}) \cdot \mathrm{vol}(B)$
3. $\phi(K) \leq O(\phi)$

The running time is $O(\frac{\mathrm{vol}(B)\log \mathrm{vol}(B)}{\phi})$.

Now we can search for the smallest ϕ that gives case (2) of Theorem 1, which must give a cut of conductance within an $O(1)$ factor of the best we can hope for (i.e., $\phi(B)$). If we search with geometrically decreasing ϕ values, then the running time is $O(\mathrm{vol}(B)\log \mathrm{vol}(B)/\phi(B))$.

Theorem 2 and 3 hold due to the particular flow-based dynamics of the CRD algorithm, which tends to keep the diffusion local, without leaking mass out of a bottleneck.

Formally, for each CRD step, we can bound the total amount of mass that starts on nodes in B, and leaves B at any point during the diffusion. We have the following lemma, a sketch of the proof of which is given. We include the full proof in Appendix B.

Lemma 1. *In the j-th CRD step, let M_j be the total amount of mass in B at the start, and let L_j be the amount of mass that ever leaves B during the diffusion. Then $L_j \leq O(\frac{1}{\sigma_2 \log \mathrm{vol}(B)}) \cdot M_j$, when $M_j \leq \mathrm{vol}_B(B)/2$; and $L_j \leq O(\frac{1}{\sigma_1}) \cdot M_j$, when $M_j \geq \mathrm{vol}_B(B)/2$.*

Proof sketch. We have two cases, corresponding to whether the diffusion already spread a lot of mass over B.

In the first case, if $M_j \geq \mathrm{vol}_B(B)/2$, then we use the upper bound $1/\phi$ that is enforced on the net mass over any edge to limit the amount of mass that can leak out. In particular $L_j \leq O(\mathrm{vol}(B)\phi(B)/\phi_S(B))$, since there are $\mathrm{vol}(B)\phi(B)$ edges from B to $\bar{B}$, and $\phi = \Theta(\phi_S(B))$ in *CRD-inner*. As $M_j \geq \Omega(\mathrm{vol}(B))$, we have $L_j \leq O(\frac{1}{\sigma_1}) \cdot M_j$.

The second case is when $M_j \leq \mathrm{vol}_B(B)/2$. In this case, a combination of Assumption 2 and capacity releasing controls the leakage of mass. Intuitively, there are still many nodes in B to which the diffusion can spread mass. For the nodes in B with excess on them, when they push their excess, most of the downhill directions go to nodes inside B. As a consequence of capacity releasing, only a small fraction of mass will leak out. □

Theorem 2 and 3 follow from straightforward analysis of the total amount of leaked mass at termination. We sketch the ideas for the proof of Theorem 2, with the full proof in Appendix B.

Proof sketch. Since we use $\phi = \Theta(\phi_S(B))$ when we call *CRD-inner*, the diffusion will be able to spread mass over nodes inside B, since there is no bottleneck with conductance smaller than $\phi_S(B)$ in B.

Thus, before every node v in B has $d(v)$ mass on it (in which case we say v is *saturated*), there will be no excess on nodes in B at the end of a CRD step. Consequently, the amount of mass in B only decreases (compared to the supposed $2d(v_s) \cdot 2^j$ amount in the j-th CRD step) due to mass leaving B.

As long as the total amount M_j of mass in B at the start of a CRD step is less than $\mathrm{vol}_B(B)/2$, the mass loss to $\bar{B}$ is at most a $O(1/(\sigma_2 \log \mathrm{vol}(B)))$ fraction of the mass in B each CRD step. After $O(\log \mathrm{vol}(B))$ CRD steps, M_j reaches $\mathrm{vol}(B)_B/2$, and only a $O(1/\sigma_2)$ fraction of mass has left B so far. After $O(1)$ more CRD steps, there will be enough mass to saturate all nodes in B, and each of these CRD steps looses at most a $O(1/\sigma_1)$ fraction of the mass to $\bar{B}$. Thus we loose at most a $O(1/\sigma)$ fraction of mass before all nodes in B are saturated.

Once the diffusion has saturated all nodes in B, the amount of mass in B will be $2\mathrm{vol}(B)$ at the start of every subsequent CRD step. At most $\mathrm{vol}(B)\phi(B)/\phi_S(B) \leq O(\mathrm{vol}(B)/\sigma)$ mass can leave B, and nodes in B can hold $\mathrm{vol}(B)$ mass, so there must be a lot of excess (in B) at the end. Thus, the CRD algorithm will terminate in at most 2 more CRD steps, since the amount of mass almost stops growing due to excess removal.

At termination, the amount of mass is $\Theta(\mathrm{vol}(B))$, and only $O(1/\sigma)$ fraction of the mass is in $\bar{B}$. Since $S = \{v | m(v) \geq d(v)\}$, and the total mass outside is $O(\mathrm{vol}(B)/\sigma)$, we get claim (1) of the theorem. In our simplified argument, all nodes in B have saturated sinks (i.e., $\mathrm{vol}(B \setminus S) = 0$) at termination. We get the small loss in claim (2) when we carry out the argument in more detail.

The amount of mass grows geometrically before the CRD algorithm terminates, so the running time is dominated by the last CRD step. The total amount of mass is $O(\mathrm{vol}(B))$ in the last CRD step, and the running time follows Theorem 1 with $\phi = \Theta(\phi_S(B))$. □

The proof of Theorem 3 is very similar to Theorem 2, and the conductance guarantee follows directly from Theorem 1.

4. Empirical Illustration

We have compared the performance of the CRD algorithm (Algorithm 1), the Andersen-Chung-Lang local spec-

Table 1. Ground truth clusters

graph	feature	volume	nodes	cond.
Hop.	217	10696	200	0.26
	2009	32454	886	0.19
Rice	203	43321	403	0.46
	2009	30858	607	0.33
Sim.	2007	14424	281	0.47
	2009	11845	277	0.1
Colgate	2006	62064	556	0.48
	2007	68381	588	0.41
	2008	62429	640	0.29
	2009	35369	638	0.11

tral algorithm (ACL) (2006), and the flow-improve algorithm (FlowImp) (Andersen & Lang, 2008). Given a starting node v_s and teleportation probability α, ACL is a local algorithm that computes an approximate personalized PageRank vector, which is then used to identify local structure via a sweep cut. FlowImp is a flow-based algorithm that takes as input a set of reference nodes and finds a cluster around the given reference set with small conductance value. Note that we only couple FlowImp with ACL. The reason is that, while FlowImp needs a very good reference set as input to give meaningful results in our setting, it can be used as a "clean up" step for spectral methods, since they give good enough output. Note also that FlowImp has running time that depends on the volume of the entire graph, as it optimizes a global objective, while our CRD algorithm takes time linear in the volume of the local region explored.

We compare these methods on 5 datasets, one of which is a synthetic grid graph. For the 4 real-world graphs, we use the Facebook college graphs of John Hopkins (Hop.), Rice, Simmons (Sim.), and Colgate, as introduced in Traud et al. (2012). Each graph in the Facebook dataset comes along with some features, e.g., "dorm 217," and "class year 2009." We consider a set of nodes with the same feature as a "ground truth" cluster, e.g., students of year 2009. We filter out very noisy features via some reasonable thresholds, and we run our computations on the the remaining features. We include the details of the graph and feature selection in Appendix C. The clusters of the features we use are shown in Table 1.

We filter bad clusters from all the ground truth clusters, by setting reasonable thresholds on volume, conductance, and gap (which is the ratio between the spectral gap of the induced graph of cluster, and the cut conductance of the cluster). Details about the selection of the ground truth clusters are in Appendix refsxn:empirical-appendix. In Table 1, we show the size and conductance of the clusters of the features used in our experiments.

For the synthetic experiment, we measure performance by conductance; the smaller the better. For real-world experiments, we use precision and recall. We also compare to ACLopt which "cheats" in the sense that *it uses ground truth* to choose the parameter α with best F1-score (a combination of precision and recall). A detailed discussion on parameter tuning of the algorithms is given in Appendix C.

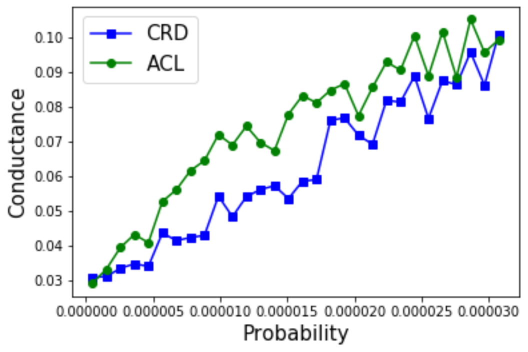

Figure 4. Average results for 60×60 grid

Table 2. Results on Facebook graphs

ID	feat.	mtr.	CRD	ACL	ACLopt	FlowImp
Hopkins	217	Pr.	0.92	0.87	0.87	0.92
		Re.	0.95	0.94	0.94	0.89
	2009	Pr.	0.95	0.92	0.91	0.96
		Re.	0.97	0.95	0.95	0.96
Rice	203	Pr.	0.43	0.32	0.32	0.33
		Re.	0.8	0.9	0.9	0.87
	2009	Pr.	0.92	0.25	0.25	0.92
		Re.	0.98	0.99	0.99	0.99
Simmons	2007	Pr.	0.5	0.49	0.49	0.26
		Re.	0.5	0.75	0.75	0.99
	2009	Pr.	0.96	0.95	0.95	0.96
		Re.	0.99	0.99	0.99	0.99
Colgate	2006	Pr.	0.43	0.41	0.41	0.22
		Re.	0.53	0.68	0.69	1.0
	2007	Pr.	0.52	0.47	0.47	0.24
		Re.	0.57	0.71	0.72	1.0
	2008	Pr.	0.94	0.61	0.64	0.95
		Re.	0.96	0.95	0.95	0.97
	2009	Pr.	0.97	0.93	0.93	0.98
		Re.	0.98	0.98	0.98	0.99

For the synthetic data, we use a grid graph of size 60×60. We add noise to the grid by randomly connecting two vertices. We illustrate the performance of the algorithms versus probability of random connection in Figure 4. The range of probabilities was chosen consistent with theory. As expected, CRD outperforms ACL in the intermediate range, and the two method's performances meet at the endpoints. One view of this is that the random connections initially adds noise to the local structure and eventually destroys it. CRD is more tolerant to this noise process.

See Table 2 for results for real-world data. We run algorithms starting at each vertex in a random sample of half the vertices in each cluster and report the median.

For clusters with good but not great conductance (e.g., Rice 2009, Colgate 2008), CDR outperforms ACL and has nearly identical performance to FlowImp (which, recall, is a global algorithm). This is a consequence of CDR avoiding the trap of leaking mass out of the local structure, in contrast to ACL, which leaks a large fraction of mass. For clusters with great conductance, all methods perform very well; and all methods perform poorly when the conductance of the clusters gets close to 0.5. We include more detailed plots and discussion in Appendix C.

Here again, as with the synthetic data, we see that for high conductance sets (which do not have good local structure) and very good conductance sets (which have excellent local structure), all methods perform similarly. In the intermediate range, i.e., when there are moderately good but not very good quality clusters, CDR shows distinct advantages, as suggested by the theory.

Acknowledgements

SR and DW are supported by the National Science Foundation under Grant CCF-1528174 and CCF-1535989. MM and KF would like to thank the Army Research Office and the Defense Advanced Research Projects Agency for partial support of this work. MH has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement no. 340506.

References

Alon, Noga, Benjamini, Itai, Lubetzky, Eyal, and Sodin, Sasha. Non-backtracking random walks mix faster. *Communications in Contemporary Mathematics*, 9(04): 585–603, 2007.

Andersen, R. and Lang, K. An algorithm for improving graph partitions. *SODA 2008*, pp. 651–660, 2008.

Andersen, Reid and Peres, Yuval. Finding sparse cuts locally using evolving sets. In *STOC 2009*, pp. 235–244, 2009.

Andersen, Reid, Chung, Fan, and Lang, Kevin. Local graph partitioning using PageRank vectors. In *FOCS 2006*, pp. 475–486, 2006.

Arora, Sanjeev, Rao, Satish, and Vazirani, Umesh. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Cheeger, Jeff. A lower bound for the smallest eigenvalue of the Laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*, 1969.

Donath, William E and Hoffman, Alan J. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

Gleich, D. F. PageRank beyond the web. *SIAM Review*, 57 (3):321–363, 2015.

Goldberg, Andrew V and Tarjan, Robert E. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

Goldberg, Andrew V. and Tarjan, Robert Endre. Efficient maximum flow algorithms. *Commun. ACM*, 57(8):82–89, 2014.

Guattery, S. and Miller, G.L. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19:701–719, 1998.

Jeh, Glen and Widom, Jennifer. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279. ACM, 2003.

Jeub, L. G. S., Balachandran, P., Porter, M. A., Mucha, P. J., and Mahoney, M. W. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91:012821, 2015.

Leighton, Tom and Rao, Satish. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *FOCS, 1988*, pp. 422–431. IEEE, 1988.

Leskovec, J., Lang, K.J., Dasgupta, A., and Mahoney, M.W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

Mahoney, M. W., Orecchia, L., and Vishnoi, N. K. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *Journal of Machine Learning Research*, 13:2339–2365, 2012.

Ng, A.Y., Jordan, M.I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *NIPS '01: Proceedings of the 15th Annual Conference on Advances in Neural Information Processing Systems*, 2001.

Orecchia, Lorenzo, Sachdeva, Sushant, and Vishnoi, Nisheeth K. Approximating the exponential, the Lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. In *STOC 2012*, pp. 1141–1160. ACM, 2012.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Transcations of Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Shun, J., Roosta-Khorasani, F., Fountoulakis, K., and Mahoney, M. W. Parallel local graph clustering. *Proceedings of the VLDB Endowment*, 9(12):1041–1052, 2016.

Spielman, Daniel A. and Teng, Shang-Hua. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC 2004*, pp. 81–90, 2004.

Traud, A. L., Mucha, P. J., and Porter, M. A. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

von Luxburg, U. A tutorial on spectral clustering. Technical Report 149, Max Plank Institute for Biological Cybernetics, August 2006.

White, S. and Smyth, P. A spectral clustering approach to finding communities in graphs. In *SDM '05: Proceedings of the 5th SIAM International Conference on Data Mining*, pp. 76–84, 2005.

Zhu, Zeyuan Allen, Lattanzi, Silvio, and Mirrokni, Vahab S. A local algorithm for finding well-connected clusters. In *ICML 2013*, pp. 396–404, 2013.

A. CRD Inner Procedure

We first fill in the missing details in the *CRD-inner* subroutine (Algorithm 3).

Note an ineligible arc (v,u) must remain ineligible until the next relabel of v, so we only need to check each arc out of v once between consecutive relabels. We use $\mathrm{current}(v)$ to keep track of the arcs out of v that we have checked since the last relabel of v. We always pick an active vertex v with the lowest label. Then for any eligible arc (v,u), we know $m(u) \le d(u)$, so we can push at least 1 along (v,u) (without violating $m(u) \le 2d(u)$), which is crucial to bound the total work.

We keep the list Q in non-decreasing order of the vertices' labels, for efficient look-up of the lowest labeled active vertex, and *Add, Remove, Shift* are the operations to maintain this order. Note these operations can be implemented to take $O(1)$ work. In particular, when we add a node u to Q, it will always be the active node with lowest label, so will be put at the beginning. We only remove the first element v from Q, and when we shift a node v in Q, we know $l(v)$ increases by exactly 1. To maintain Q, we simply need to pick two linked lists, one containing all the active nodes with non-decreasing labels, and another linked list containing one pointer for each label value, as long as there is some

Algorithm 3 *CRD-inner*$(G,m(\cdot),\phi)$

```
. Initialization:
. . ∀$\{v,u\} \in E$, $m(u,v) = m(v,u) = 0$.
. . $Q = \{v | m(v) > d(v)\}$, $h = \frac{3\log|m(\cdot)|}{\phi}$
. . ∀$v$, $l(v) = 0$, and $\mathrm{current}(v)$ is the first edge in
      $v$'s list of incident edges.
. While $Q$ is not empty
. . Let $v$ be the lowest labeled vertex in $Q$.
. . Push/Relabel$(v)$.
. . If Push/Relabel$(v)$ pushes mass along $(v,u)$
. . . If $v$ becomes in-active, Remove$(v,Q)$
. . . If $u$ becomes active, Add$(u,Q)$
. . Else If Push/Relabel$(v)$ increases $l(v)$ by 1
. . . If $l(v) < h$, Shift$(v,Q)$
. . . Else Remove$(v,Q)$

Push/Relabel$(v)$
. Let $\{v,u\}$ be $\mathrm{current}(v)$.
. If arc $(v,u)$ is eligible, then Push$(v,u)$.
. Else
. . If $\{v,u\}$ is not the last edge in $v$'s list of edges.
. . . Set $\mathrm{current}(v)$ be the next edge of $v$.
. . Else (i.e., $\{v,u\}$ is the last edge of $v$)
. . . Relabel$(v)$, and set $\mathrm{current}(v)$ be the first
        edge of $v$'s list of edges.

Push$(v,u)$
. Assertion: $r_m(v,u) > 0, l(v) \ge l(u) + 1$.
             $\mathrm{ex}(v) > 0, m(u) < 2d(u)$.
. $\psi = \min(\mathrm{ex}(v), r_m(v,u), 2d(u) - m(u))$
. Send $\psi$ units of mass from $v$ to $u$:
   $m(v,u) \leftarrow m(v,u) + \psi, m(u,v) \leftarrow m(u,v) - \psi$.
   $m(v) \leftarrow m(v) - \psi, m(u) \leftarrow m(u) + \psi$.

Relabel$(v)$
. Assertion: $v$ is active, and ∀$u \in V$,
                 $r_m(v,u) > 0 \implies l(v) \le l(u)$.
. $l(v) \leftarrow l(v) + 1$.
```

active node with that label, and the pointer contains the position of first such active node in Q. Maintaining this two lists together can give $O(1)$ time *Add, Remove, Shift*.

Now we proceed to prove the main theorem of *CRD-inner*.

Theorem 1. *Given $G, m(\cdot)$, and $\phi \in (0,1]$, such that $|m(\cdot)| \leq \mathrm{vol}(G)$, and $\forall v : m(v) \leq 2d(v)$ at the start,* CRD-inner *terminates with one of the following cases:*

1. CRD-inner *finishes the full CRD step: $\forall v : m(v) \leq d(v)$.*
2. *There are nodes with excess, and we can find a cut A of conductance $O(\phi)$. Moreover, $\forall v \in A : 2d(v) \geq m(v) \geq d(v)$, and $\forall v \in \bar{A} : m(v) \leq d(v)$.*

The running time is $O(|m(\cdot)| \log(|m(\cdot)|)/\phi)$.

Proof. Let $l(\cdot)$ be the labels of vertices at termination, and let $B_i = \{v | l(v) = i\}$. We make the following observations: $l(v) = h \Rightarrow 2d(v) \geq m(v) \geq d(v)$; $h > l(v) \geq 1 \Rightarrow m(v) = d(v)$; $l(v) = 0 \Rightarrow m(v) \leq d(v)$.

Since $|m(\cdot)| \leq \mathrm{vol}(G)$, if $B_0 = \emptyset$, it must be $|m(\cdot)| = \mathrm{vol}(G)$, and every v has $m(v) = d(v)$, so we get case (1). If $B_h = \emptyset$, we also get case (1).

If $B_h, B_0 \neq \emptyset$, let $S_i = \cup_{j=i}^{h} B_j$ be the set of nodes with label at least i. We have h level cuts $S_h, \ldots, S_1$, where $\mathrm{vol}(S_h) \geq 1$, and $S_j \subseteq S_i$ if $j > i$. We claim one of these level cuts must have conductance $O(\phi)$. For any S_i, we divide the edges from S_i to $\overline{S_i}$ into two groups: 1) edge across one level (i.e., from node in B_i to node in B_{i-1}), and 2) edges across more than one level. Let $z_1(i), z_2(i)$ be the number of edges in the two groups respectively, and define $\phi_g(i) \overset{\mathsf{def}}{=} z_g(i)/\mathrm{vol}(S_i)$ for $g = 1, 2$.

First we show that, there must be a i^* between h and $h/2$ such that $\phi_1(i^*) \leq \phi$. By contradiction, if $\phi_1(i) > \phi$ for all $i = h, \ldots, h/2$, since $\mathrm{vol}(S_{i-1}) \geq \mathrm{vol}(S_i)(1 + \phi_1(S_i))$, we get $\mathrm{vol}(S_{h/2}) \geq (1+\phi)^{h/2}\mathrm{vol}(S_h)$. With $h = 3\log|m(\cdot)|/\phi$, we have $\mathrm{vol}(S_{h/2}) \geq \Omega(|m(\cdot)|^{3/2})$, and since nodes in $S_{h/2}$ are all saturated, we get a contradiction since we must have $\mathrm{vol}(S_{h/2}) \leq |m(\cdot)|$.

Now we consider any edge $\{v, u\}$ counted in $z_2(i^*)$ (i.e., $v \in S_{i^*}, u \in \overline{S_{i^*}}, l(v) - l(u) \geq 2$). Since $i^* \geq h/2 > 1/\phi$, $\hat{c}(v,u) = 1/\phi$. $l(v) - l(u) > 2$ suggests $r_m(v,u) = 0$, thus $m(v,u) = 1/\phi$ (i.e., $1/\phi$ mass pushed out of S_{i^*} along each edge counted in $z_2(i^*)$). Each edge counted in $z_1(i^*)$ can have at most $1/\phi$ mass pushed into S_{i^*}, and at most $2\mathrm{vol}(S_{i^*})$ mass can start in S_{i^*}, then we know

$$z_2(i^*)/\phi \leq z_1(i^*)/\phi + 2\mathrm{vol}(S_{i^*})$$

We will let A be S_{i^*}, and we have

$$\phi(A) = \frac{z_1(i^*) + z_2(i^*)}{\mathrm{vol}(S_{i^*})} \leq 4\phi = O(\phi)$$

Here we assume S_{i^*} is the smaller side of the cut to compute the conductance. If this is not the case, i.e. $\mathrm{vol}(S_{i^*}) > \mathrm{vol}(G)/2$, we just carry out the same argument as above, but run the region growing argument from level $h/4$ up to level $h/2$, and get a low conductance cut, and still let A to be the side containing S_h. The additional properties of elements in A follows from $S_h \subseteq A \subseteq S_{h/4}$.

Now we proceed to the running time. The initialization takes $O(|m(\cdot)|)$. Subsequently, each iteration takes $O(1)$ work. We will first attribute the work in each iteration to either a push or a relabel. Then we will charge the work on pushes and relabels to the absorbed mass, such that each unit of absorbed mass gets charged $O(h)$ work. Recall the absorbed mass at v are the first up to $d(v)$ mass starting at or pushed into v, and these mass never leave v, as the algorithm only pushes excess mass. This will prove the result, as there are at most $|m(\cdot)|$ units of (absorbed) mass in total.

In each iteration of *Unit-Flow*, the algorithm picks a lowest labeled active node v. If *Push/Relabel*(v) ends with a push of ψ mass, we charge $O(\psi)$ to that push operation. Since $\psi \geq 1$, we charged the push enough to cover the work in that iteration. If the call to *Push/Relabel*(v) doesn't push, we charge the $O(1)$ work of the iteration to the next relabel of v (or the last relabel if there is no next relabel). The latter can happen at most $d(v)$ times between consecutive relabels of v, so each relabel of v is charged $O(d(v))$ work.

We now charge the work on pushes and relabels to the absorbed mass. Note each time we relabel v, there are $d(v)$ units of absorbed mass at v, so we charge the $O(d(v))$ work on the relabel to the absorbed mass, and each unit gets charged $O(1)$. There is at most h relabels of v, so each unit of absorbed mass is charged $O(h)$ in total by all the relabels.

For the work on pushes, we consider the potential function $\Lambda = \sum_v \mathrm{ex}(v)l(v)$. Λ is always non-negative, and as we only push excess mass downhill, each push of ψ units of mass decrease Λ by at least ψ, so we can charge the work on pushes to the increment of Λ. It only increases at relabel. When we relabel v, Λ is increased by $\mathrm{ex}(v)$. Since $\mathrm{ex}(v) \leq d(v)$, we can charge $O(1)$ to each unit of absorbed mass at v to cover Λ's increment. In total we can charge all pushes (via Λ) to absorbed mass, and each unit is charged with $O(h)$.

If we need to compute the cut A in case (2), the running time is $O(\mathrm{vol}(S_1))$, which is $O(|m(\cdot)|)$. $\square$

B. Local Clustering

Recall we assume B to satisfy the following conditions.

Assumption 1. $\sigma_1 \overset{\text{def}}{=} \frac{\phi_S(B)}{\phi(B)} \geq \Omega(1)$.

Assumption 2. *There exists $\sigma_2 \geq \Omega(1)$, such that any $T \subset B$ with $vol_B(T) \leq \mathrm{vol}_B(B)/2$ satisfies*

$$\frac{|E(T, B \setminus T)|}{|E(T, V \setminus B)| \log \mathrm{vol}(B) \log \frac{1}{\phi_S(B)}} \geq \sigma_2.$$

Now we proceed to prove the main lemma.

Lemma 1. *In the j-th CRD step, let M_j be the total amount of mass in B at the start, and let L_j be the amount of mass that ever leaves B during the diffusion. Then $L_j \leq O(\frac{1}{\sigma_2 \log \mathrm{vol}(B)}) \cdot M_j$, when $M_j \leq \mathrm{vol}_B(B)/2$; and $L_j \leq O(\frac{1}{\sigma_1}) \cdot M_j$, when $M_j \geq \mathrm{vol}_B(B)/2$.*

Proof. For simplicity, we assume once a unit of mass leaves B, it is never routed back. Intuitively, mass coming back into B should only help the algorithm, and indeed the results don't change without this assumption. We denote $|M_j(S)|$ as the amount of mass on nodes in a set S at the start of the *CRD-inner* call.

We have two cases, corresponding to whether the diffusion already spread a lot of mass over B. If $M_j \geq \mathrm{vol}_B(B)/2$, we use the upperbound $1/\phi$ that is enforced on the net mass over any edge to limit the amount of mass that can leak out. In particular $L_j \leq O(\mathrm{vol}(B)\phi(B)/\phi_S(B))$, since there are $\mathrm{vol}(B)\phi(B)$ edges from B to $\bar{B}$, and $\phi = \Theta(\phi_S(B))$ in *CRD-inner*. As $M_j \geq \Omega(\mathrm{vol}(B))$, we have $L_j \leq O(\frac{1}{\sigma_1}) \cdot M_j$.

The second case is when $M_j \leq \mathrm{vol}_B(B)/2$. In this case, a combination of Assumption 2 and capacity releasing controls the leakage of mass. Intuitively, there are still many nodes in B that the diffusion can spread mass to. For the nodes in B with excess on them, when they push their excess, most of the downhill directions go to nodes inside B. As a consequence of capacity releasing, only a small fraction of mass will leak out.

In particular, let $l(\cdot)$ be the labels on nodes when *CRD-inner* finishes, we consider $B_i = \{v \in B | l(v) = i\}$ and the level cuts $S_i = \{v \in B | l(v) \geq i\}$ for $i = h, \ldots, 1$. As $M_j \leq \mathrm{vol}_B(B)/2$, we know $\mathrm{vol}(S_h) \leq \mathrm{vol}(S_{h-1}) \leq \ldots \leq \mathrm{vol}(S_1) \leq \mathrm{vol}_B(B)/2$. In this case, we can use Assumption 2 on all level cuts $S_h, \ldots, S_1$. Moreover, for a node $v \in B_i$, the "'effective"' capacity of an arc from v to $\bar{B}$ is $\min(i, 1/\phi)$. Formally, we can bound L_j by the total (effective) outgoing capacity, which is

$$\sum_{i=1}^{h} |E(B_i, \bar{B})| \cdot \min(i, \frac{1}{\phi}) = \sum_{i=1}^{\frac{1}{\phi}} |E(S_i, \bar{B})| \qquad (1)$$

where h is the bound on labels used in unit flow.

We design a charging scheme to charge the above quantity (the right hand side) to the mass in $\Delta_j(B)$, such that each unit of mass is charged $O(1/(\sigma_2 \log \mathrm{vol}(B)))$. It follows that $L_j \leq O(\frac{1}{\sigma_2 \log \mathrm{vol}(B)}) \cdot |\Delta_j(B)|$.

Recall that, $|E(S_i, \bar{B})| \leq \frac{|E(S_i, B \setminus S_i)|}{\sigma_2 \log \mathrm{vol}(B) \log(1/\phi)}$ from Assumption 2. We divide edges in $E(S_i, B \setminus S_i)$ into two groups: : 1) edges across one level, and 2) edges across more than one level. Let $z_1(i), z_2(i)$ be the number of edges in the two groups respectively.

If $z_1(i) \geq |E(S_i, B \setminus S_i)|/3$, we charge $3/(\sigma_2 \log \mathrm{vol}(B) \log(1/\phi))$ to each edge in group 1. These edges in turn transfer the charge to the absorbed mass at their endpoints in B_i. Since each node v in level $i \geq 1$ has $d(v)$ absorbed mass, each unit of absorbed mass is charged $O(1/(\sigma_2 \log \mathrm{vol}(B) \log(1/\phi)))$. Note that the group 1 edges of different level i's are disjoint, so each unit of absorbed mass will only be charged once this way.

If $z_1(i) \leq |E(S_i, B \setminus S_i)|/3$, we know $z_2(i) - z_1(i) \geq |E(S_i, B \setminus S_i)|/3$. Group 2 edges in total send at least $(i-1)z_2(i)$ mass from S_i to $B \setminus S_i$, and at most $(i-1)z_1(i)$ of these mass are pushed into S_i by group 1 edges. Thus, there are at least $(i-1)|E(S_i, B \setminus S_i)|/3$ mass that start in S_i, and are absorbed by nodes at level below i (possibly outside B). In particular, this suggests $|M_j(S_i)| \geq (i-1)|E(S_i, B \setminus S_i)|/3$, and we split the total charge $|E(S_i, \bar{B})|$ evenly on these mass, so each unit of mass is charged $O(1/(i\sigma_2 \log \mathrm{vol}(B) \log(1/\phi)))$. Since we sum from $i = 1/\phi$ to 1 in (RHS of) Eqn (1), we charge some mass multiple times (as S_i's not disjoint), but we can bound the total charge by $\sum_{i=1}^{1/\phi} \frac{1}{i} \cdot O(1/(\sigma_2 \log \mathrm{vol}(B) \log(1/\phi)))$, which is $O(1/(\sigma_2 \log \mathrm{vol}(B)))$. This completes the proof. □

Now we fill in some details for the proof of Theorem 2.

Theorem 2. *Starting from any $v_s \in B$, with the above parameters, when the CRD algorithm terminates, if we let $S = \{v | m(v) \geq d(v)\}$, then we have:*

1. $\mathrm{vol}(S \setminus B) \leq O(\frac{1}{\sigma}) \cdot \mathrm{vol}(B)$
2. $\mathrm{vol}(B \setminus S) \leq O(\frac{1}{\sigma}) \cdot \mathrm{vol}(B)$

where $\sigma = \min(\sigma_1, \sigma_2) \geq \Omega(1)$, with the σ_1, σ_2 from Assumption 1 and 2. The running time is $O(\frac{\mathrm{vol}(B) \log \mathrm{vol}(B)}{\phi_S(B)})$.

Proof. Since we use $\phi = \Theta(\phi_S(B))$ when we call *CRD-inner*, we are guaranteed that *CRD-inner* will make enough effort to get through bottleneck of conductance $\Omega(\phi_S(B))$, so the diffusion should be able to spread mass completely over B, since any cut inside B has conductance at least $\phi_S(B)$. Thus, there should be no excess remaining on nodes in B at the end of *CRD-inner*, unless every node v in B has $m(v)$ mass already (i.e., all nodes in B are *saturated*).

Formally, consider the proof of Theorem 1, but with everything with respect to the induced graph of B. If there is no excess pushed into B from outside, we can use the exact arguments in the proof of Theorem 1 to show that either there is no excess on any node in B at the end of the *CRD-inner* call, or there is a cut of conductance $O(\phi)$, or all nodes in B are saturated. By assumption, the second case is not possible, so we won't remove excess supply between *CRD-inner* calls before all nodes in B are saturated. If we consider supply pushed back into B from outside, we can show that the amount of excess on nodes in B at the end is at most the amount of mass pushed into B. Since we already counted the mass leaving B as lost, we don't need to worry about them again when we remove the mass.

Consequently, before all nodes in B are saturated, the amount of mass in B only decreases (compared to the supposed $2d(v_s) \cdot 2^j$ amount in iteration j) due to mass leaving B, which we can bound (Lemma 1) by a $O(1/(\sigma \log \text{vol}(B)))$ fraction of the mass in B each iteration. We will have enough mass to spread over all nodes in B in $O(\log \text{vol}(B))$ iterations, so we lose $O(1/\sigma)$ fraction of mass before all nodes in B are saturated.

Once the diffusion has saturated all nodes in B, the amount of mass in B will be $2\text{vol}(B)$ at the start of every subsequent *CRD-inner* call. At most $\text{vol}(B)\phi(B)/\phi_S(B) \leq O(\text{vol}(B)/\sigma)$ mass can leave B, and nodes in B can hold $\text{vol}(B)$ mass, so there must be a lot of excess (in B) at the end. Thus, the CRD algorithm will terminate in at most 2 more iterations after all nodes in B are saturated, since the amount of mass almost stops growing due to excess removal.

At termination, the amount of mass is $\Theta(\text{vol}(B))$, and only $O(1/\sigma)$ fraction of the mass is in $\bar{B}$. Since $S = \{v|m(v) \geq d(v)\}$, and the total mass outside is $O(\text{vol}(B)/\sigma)$, we get claim (1) of the theorem. In our simplified argument, all nodes in B have saturated sinks (i.e., $\text{vol}(B \setminus S) = 0$) at termination. We get the small loss in claim (2) when we carry out the argument more rigorously.

The amount of mass grows geometrically before the CRD algorithm terminates, so the running time is dominated by the last *CRD-inner* call. The total amount of mass is $O(\text{vol}(B))$ in the last iteration, and the running time follows Theorem 1 with $\phi = \Theta(\phi_S(B))$ □

C. Empirical Set-up and Results

C.1. Datasets

We chose the graphs of John Hopkins, Rice, Simmons and Colgate universities/colleges. The actual IDs of the graphs in Facebook100 dataset are Johns_Hopkins55, Rice31, Simmons81 and Colgate88. These graphs are anonymized Facebook graphs on a particular day in September 2005 for student social networks. The graphs are unweighted and they represent "friendship ties". The data form a subset of the Facebook100 dataset from (Traud et al., 2012). We chose these 4 graphs out of 100 due to their large assortativity value in the first column of Table A.2 in (Traud et al., 2012), where the data were first introduced and analyzed. Details about the graphs are shown is Table 3.

Graph	volume	nodes	edges
John Hopkins	373144	5157	186572
Rice	369652	4083	184826
Simmons	65968	1510	32984
Colgate	310086	3482	155043

Table 3. Graphs used for experiments.

Each graph in the Facebook dataset comes along with 6 features, i.e., second major, high school, gender, dorm, major index and year. We construct "ground truth" clusters by using the features for each node. In particular, we consider nodes with the same value of a feature to be a cluster, e.g., students of year 2009. We loop over all possible clusters and consider as ground truth the ones that have volume larger than 1000, conductance smaller than 0.5 and gap larger than 0.5. Filtering results in moderate scale clusters for which the internal volume is at least twice as much as the volume of the edges that leave the cluster. Additionally, gap at least 0.5 means that the smallest nonzero eigenvalue of the normalized Laplacian of the subgraph defined by the cluster is at least twice larger than the conductance of the cluster in the whole graph. The clusters per graph that satisfy the latter constraints are shown in Table 1.

We resort to social networks as our motivation is to test our algorithm against "noisy" clusters, and for social networks it is well known that reliable ground truth is only weakly related to good conductance clusters, and thus certain commonly-used notions of ground truth would not provide falsifiable insight into the method (Jeub et al., 2015). As analyzed in the original paper that introduced these datasets (Traud et al., 2012), only year and dorm features give non-trivial "assortativity coefficients", which is a "local measure of homophily". This agrees with the ground truth clusters we find, which also correspond to features of year and dorm.

C.2. Performance criteria and parameter tuning

For real-world Facebook graphs since we calculate the ground truth clusters in Table 1 then we measure performance by calculating precision and recall for the output clusters of the algorithms.

We set the parameters of CRD to $\phi = 1/3$ for all experiments. At each iteration we use sweep cut on the labels returned by the *CRD-inner* subroutine to find a cut of small conductance, and over all iterations of CRD we return the cluster with the lowest conductance.

ACL has two parameters, the teleportation parameter α and a tolerance parameter ϵ. Ideally the former should be set according to the reciprocal of the mixing time of a a random walk within the target cluster, which is equal to the smallest nonzero eigenvalue of the normalized Laplacian for the subgraph that corresponds to the target cluster. Let us denote the eigenvalue with λ. In our case the target cluster is a ground truth cluster from Table 1. We use this information to set parameter α. In particular, for each node in the clusters in Table 1 we run ACL 4 times where α is set based on a range of values in $[\lambda/2, 2\lambda]$ with a step of $(2\lambda - \lambda/2)/4$. The tolerance parameter ϵ is set to 10^{-7} for all experiments in order to guarantee accurate solutions for the PageRank linear system. For each parameter setting we use sweep cut to find a cluster of low conductance, and over all parameter settings we return the cluster with the lowest conductance value as an output of ACL.

For real-world experiments we show results for ACLopt. In this version of ACL, for each parameter setting of α we use sweep cut algorithm to obtain a low conductance cluster and then we compute its precision and recall. Over all parameter settings we keep the cluster with the best F1-score; a combination of precision and recall. This is an extra level of supervision for the selection of the teleportation parameter α, which is not possible in practice since it requires ground truth information. However, the performance of ACLopt demonstrates the performance of ACL in case that we could make optimal selection of parameter α among the given range of parameters (which also includes ground truth information) for the precision and recall criteria.

Finally, we set the reference set of FlowI to be the output set of best conductance of ACL out of its 4 runs for each node. By this we aim to obtain an improved cluster to ACL in terms of conductance. Note that FlowI is a global algorithm, which means that it accesses the information from the whole graph compared to CRD and ACL which are local algorithms.

C.3. Real-world experiments

For clusters in Table 1 we sample uniformly at random half of their nodes. For each node we run CRD, ACL and ACL+FlowI. We report the results using box plots, which graphically summarizes groups of numerical data using quartiles. In these plot the orange line is the median, the blue box below the median is the first quartile, the blue box above the median is the third quartile, the extended long lines below and above the box are the maximum and minimum values and the circles are outliers.

The results for John Hopkins university are shown in Figure 5. Notice in this figure that CRD performs better than ACL and ACLopt, which both use ground truth information, see parameter tuning in Subsection C.2. CRD performs similarly to ACL+FlowI, where FlowI is a global algorithm, but CRD is a local algorithm. Overall all methods have large medians for this graph because the clusters with dorm 217 and year 2009 are clusters with low conductance compared to the ones in other universities/colleges which we will discuss in the remaining experiments of this subsection.

The results for Rice university are shown in Figure 6. Notice that both clusters of dorm 203 and year 2009 for Rice university are worse in terms of conductance compared to the clusters of John Hopkins university. Therefore the performance of the methods is decreased. For the cluster of dorm 203 with conductance 0.46 CRD has larger median than ACL, ACLopt and ACL+Flow in terms of precision. The latter methods obtain larger median for recall, but this is because ACL leaks lots of probability mass outside of the ground truth cluster since as indicated by its large conductance value many nodes in this cluster are connected externally. For cluster of year 2009 CRD outperforms ACL, which fails to recover the cluster because it leaks mass outside the cluster, FlowI corrects the problem and locates the correct cluster at the expense of touching the whole graph. Notice that all methods have a significant amount of variance and outliers, which is also explained by the large conductance values of the clusters.

The results for Simmons college are shown in Figure 7. Notice that Simmons college in Table 1 has two clusters, one with poor conductance 0.47 for students of year 2007 and one low conductance 0.1 for students of year 2009. The former with conductance 0.47 means that the internal volume is nearly half the volume of the outgoing edges. This has a strong implication in the performance of CRD, ACL and ACLopt which get median precision about 0.5. This happens because the methods push half of the flow (CRD) and half of the probability mass (ACL) outside the ground truth cluster, which results in median precision 0.5. ACL achieves about 20% more (median) recall than CRD but this is because ACL touched more nodes than CRD during execution of the algorithm. Notice that ACL+FlowI fails for the cluster of year 2007, this is because FlowI is a global algorithm, hence it finds a cluster that has low conductance but it is not the ground truth cluster. The second cluster of year 2009 has low conductance hence all methods have large median performance with CRD being slightly better than ACL, ACLopt and ACL+FlowI.

The results for Colgate university are shown in Figure 8. The interesting property of the clusters in Table 1 for Colgate university is that their conductance varies from low 0.1 to large 0.48. Therefore in Figure 8 we see a smooth tran-

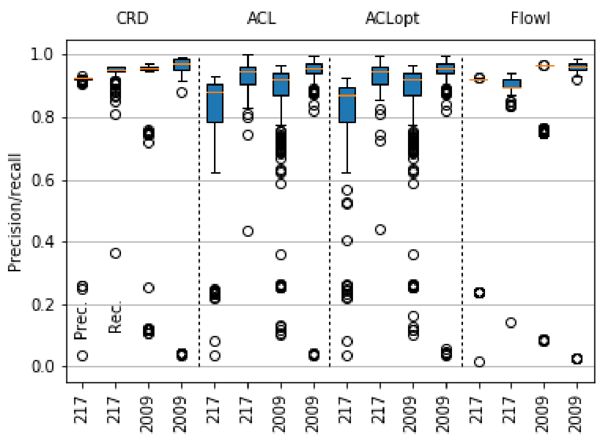

Figure 5. Precision and recall results for John Hopkins university

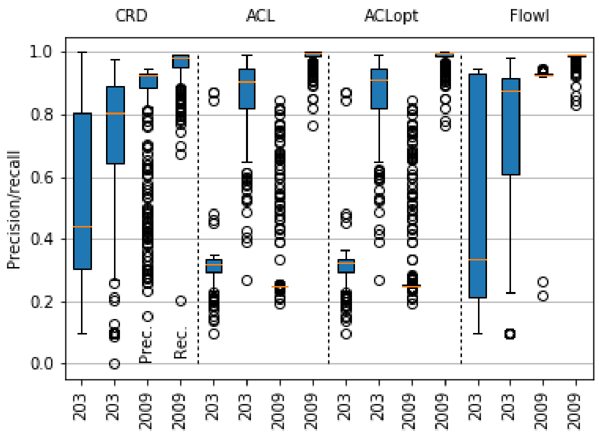

Figure 6. Precision and recall results for Rice university

sition of performance for all methods from poor to good performance. In particular, for the cluster of year 2006 the conductance is 0.48 and CRD, ACL and ACLopt perform poorly by having median precision about 50%, recall is slightly better for ACL but this is because we allow it touch a bigger part of the graph. ACL+FlowI fails to locate the cluster. For the cluster of year 2007 the conductance is 0.41 and the performance of CRD, ACL and ACLopt is increased with CRD having larger (median) precision and ACL having larger (median) recall as in the previous cluster. Conductance is smaller for the cluster of year 2008, for which we observe substantially improved performance for CRD with large median precision and recall. On the contrary, ACL, ACLopt and ACL+FlowI have nearly 30% less median precision in the best case and similar median recall, but only because a large amount of probability mass is leaked and a big part of the graph is touched which includes the ground truth cluster. Finally, the cluster of year 2009 has low conductance 0.11 and all methods have good performance for precision and recall.

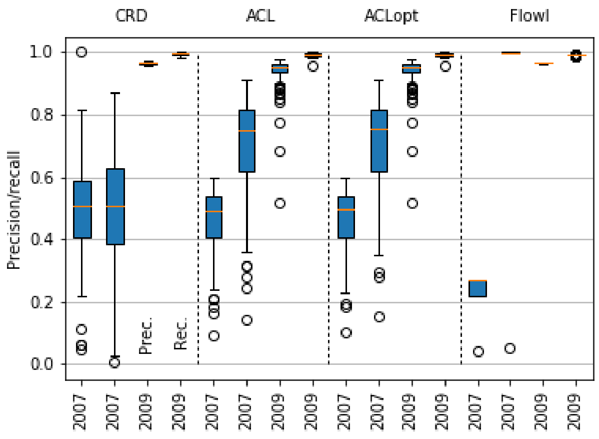

Figure 7. Precision and recall results for Simmons college

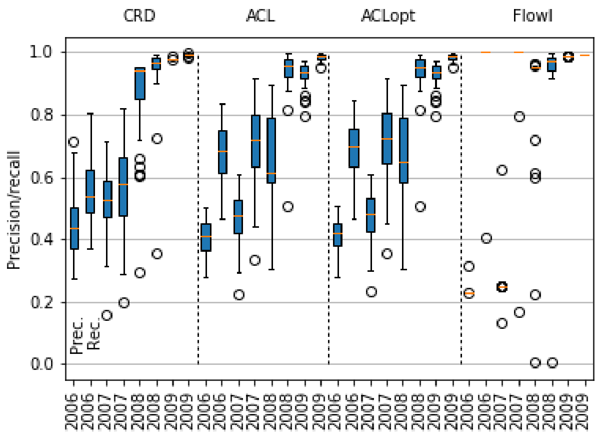

Figure 8. Precision and recall results for Colgate university