Flow-Based Local Graph Clustering with Better Seed Set Inclusion

Nate Veldt [*] Christine Klymko [†] David F. Gleich [‡]

Abstract

Flow-based methods for local graph clustering have received significant recent attention for their theoretical cut improvement and runtime guarantees. In this work we present two improvements for using flow-based methods in real-world semi-supervised clustering problems. Our first contribution is a generalized objective function that allows practitioners to place strict and soft penalties on excluding specific seed nodes from the output set. This feature allows us to avoid the tendency, often exhibited by previous flow-based methods, to contract a large seed set into a small set of nodes that does not contain all or even most of the seed nodes. Our second contribution is a fast algorithm for minimizing our generalized objective function, based on a variant of the push-relabel algorithm for computing preflows. We make our approach very fast in practice by implementing a global relabeling heuristic and employing a warm-start procedure to quickly solve related cut problems. In practice our algorithm is faster than previous related flow-based methods, and is also more robust in detecting ground truth target regions in a graph thanks to its ability to better incorporate semi-supervised information about target clusters.

1 Introduction

Local graph clustering is the task of finding tightly connected clusters of nodes nearby a set of seed vertices in a large graph. This task has been applied to solve problems in information retrieval [9], image segmentation [11, 16], and community detection [3, 8], among many other applications. In practice, seed nodes represent semi-supervised information about a hidden target cluster, and the goal is to recover or detect this cluster by combining knowledge of the seed set with observations about the topological structure of the network.

One popular approach for graph clustering is to apply flow-based methods, which repeatedly solve regionally biased minimum cut and maximum flow problems on the input graph. These methods satisfy very good theoretical cut improvement guarantees with respect to quotient-style clustering objectives such as conductance [2, 9, 13, 16]. Additionally, some of these methods are strongly local, i.e. their runtime depends only on the size of the seed set and not the entire input graph.

Despite these attractive theoretical properties, existing flow-based methods exhibit drawbacks when it comes to solving real-world graph clustering problems. In some cases these methods exhibit the tendency to discard important semi-supervised information in favor of optimizing a quotient-style clustering objective. More specifically, existing methods either shrink a seed set of nodes into a subset with better cut-to-size ratio [9], or try to find a good output cluster that overlaps well with the seed set but may not include all or even a majority of the seed nodes [2, 13, 16]. While this is beneficial for obtaining theoretically good graph cuts, it is not always desirable in label propagation and community detection applications where the goal is to grow a set of seed nodes into a larger community. In addition to this, the previously cited flow-based methods treat all seed nodes equally, whereas in practice there may be varying levels of confidence for whether or not a seed node is a true representative of the undetected target cluster.

Although recently developed strongly-local methods constitute a major advancement in flow-based clustering, these also exhibit drawbacks in terms of implementation and practical performance. The LocalImprove algorithm of Orecchia and Zhou [13] is known to have an extremely good theoretical runtime but relies on a complicated variation of Dinic's maxflow algorithm [5] that is difficult to implement in practice. In more recent work we developed an algorithm called SimpleLocal, which provides a simplified framework for optimizing the same objective [16]. While this method is easy to implement and reasonably fast in practice, it still relies on repeatedly solving numerous exact maximum flow problems, and takes no advantage of warm-start solutions between consecutive flow problems that are closely related.

Our Contributions In this paper we improve the practical performance of flow-based methods for local clustering in two major ways. We first develop a

[*]Mathematics Department, Purdue University. Email: *lveldt@purdue.edu*. Supported by NSF award CCF-1149756.

[†]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory. Email: *klymko1@llnl.gov*. Prepared by LLNL under Contract DE-AC52-07NA27344.

[‡]Dept. of Computer Science, Purdue University. Email: *dgleich@purdue.edu*. Supported by DARPA SIMPLEX and NSF award CCF-1149756, IIS-1422918, IIS-1546488, CCF093937 and the Sloan Foundation.

generalized framework which takes better advantage of semi-supervised information about target clusters, and avoids the tendency of other methods to contract a large seed set into a small subcluster. Our approach allows users to place strict constraints and soft penalties on excluding specified seed nodes from the output set, depending on the user's level of confidence for whether or not each node should belong to the output set.

Our second major contribution is a fast algorithm for minimizing our generalized objective function. We begin by showing that this objective can be minimized in strongly-local time using a meta-procedure that repeatedly solves localized minimum *s*-*t* cuts, and doesn't require any explicit computation of maximum *s*-*t* flows. This simultaneously generalizes and simplifies the meta-procedure we developed in previous work, which solves a more restricted objective function and requires the explicit computation of maximum flows as an intermediary step to obtaining minimum cuts [16]. We then implement our meta-procedure using a fast variant of the push-relabel algorithm [6]. We make our algorithm extremely efficient using two key heuristics: a known global-relabeling scheme for the push-relabel algorithm [4], and a warm-start procedure which allows us to quickly solve consecutive minimum cut problems. We validate our approach in several community detection experiments in real-world graphs and in several large-scale 3D image segmentation problems on graphs with hundreds of millions of edges. In practice our algorithm is faster that existing implementations of related flow-based methods, and is more robust in detecting target clusters.

2 Background and Related Work

Let $G = (V, E)$ represent an undirected and unweighted graph. For each node $v \in V$ let d_v be its degree, i.e. the number of edges that have v as an endpoint. For any set $S \subset V$ let $|E_S|$ be the number of interior edges in S, $\mathbf{vol}(S) = \sum_{v \in S} d_v$ be the *volume* S, and $\mathbf{cut}(S) = \mathbf{vol}(S) - 2|E_S|$ denote the number of edges crossing from S to $\bar{S} = V \backslash S$. Each set S uniquely identifies a set of edges crossing from S to $\bar{S}$, so we will frequently refer to a set of nodes S as a *cut* in a graph. One way to quantify the community structure of a set S is by measuring its conductance:

$$\phi(S) = \mathbf{cut}(S)/[\min\{\mathbf{vol}(S), \mathbf{vol}(\bar{S})\}].$$

Small $\phi(S)$ indicates that S is well-connected internally but only loosely connected to the rest of the graph, and thus represents a topologically "good" cluster.

2.1 Local Variants of Conductance

In local graph clustering we are given a seed (or *reference*) set $R \subset V$ that is small with respect to the size of the graph. If we fix some value of a *locality* parameter $\varepsilon \in \left[\frac{\mathbf{vol}(R)}{\mathbf{vol}(\bar{R})}, \infty\right)$, then the following objective function is a modification of the conductance score biased towards the set R, which we call the *local conductance* measure:

$$\phi_{R,\varepsilon}(S) = \frac{\mathbf{cut}(S)}{\mathbf{vol}(R \cap S) - \varepsilon \mathbf{vol}(\bar{R} \cap S)}. \tag{2.1}$$

One approach to local community detection is to optimize (2.1) over sets S such that the denominator is positive. This function was first introduced specifically for $\varepsilon = \mathbf{vol}(R)/\mathbf{vol}(\bar{R})$ by Andersen and Lang [2]. Orecchia and Zhou later considered larger values of ε, effectively restricting the search space to sets S that overlap significantly with R, leading to a strongly-local algorithm [13]. Both of these methods generalize the Max-flow Quotient-cut Improvement (MQI) algorithm [9], which computes the minimum conductance subset of R, and fits the above paradigm if we allow $\varepsilon = \infty$.

2.2 Minimizing Local Conductance

Although it is NP-hard to find the minimum conductance set of a graph G, one can minimize (2.1) efficiently by repeatedly solving a sequence of related minimum *s*-*t* cut problems. First fix a parameter $\alpha \in (0, 1)$ and construct a new graph G_{st} using the following steps:

- Keep original nodes in G and edges with weight 1
- Introduce source node s and sink node t
- For every $r \in R$, connect r to s with weight αd_r
- For every $j \in \bar{R}$, connect j to t with weight $\alpha\varepsilon d_j$.

The minimum *s*-*t* cut problem seeks the minimum weight of edges in G_{st} to separate the source s from the sink t. Every subset $S \subset V$ in G induces an *s*-*t* cut in G_{st} where the two sides of the cut are $\{s\} \cup S$ and $\{t\} \cup \bar{S}$. The weight of this *s*-*t* cut can be given entirely in terms of cuts and volumes of sets in G:

$$\begin{aligned}\textsc{STcut}(S) &= \mathbf{cut}(S) + \alpha\varepsilon\mathbf{vol}(\bar{R} \cap S) + \alpha\mathbf{vol}(R \cap \bar{S}) \\ &= \mathbf{cut}(S) + \alpha\varepsilon\mathbf{vol}(\bar{R} \cap S) - \alpha\mathbf{vol}(R \cap S) + \alpha\mathbf{vol}(R).\end{aligned}$$

If there exists some S such that $\textsc{STcut}(S) < \alpha\mathbf{vol}(R)$, a few steps of algebra show that this implies $\phi_{R,\varepsilon}(S) < \alpha$. Therefore, $\phi_{R,\varepsilon}(S)$ can be minimized by finding the smallest α such that the minimum *s*-*t* cut of G_{st} is exactly $\alpha\mathbf{vol}(R)$. This can be done by performing binary search over α or simply starting with $\alpha = \phi_{R,\varepsilon}(R)$ and iteratively finding minimum *s*-*t* cuts in G_{st} for increasingly smaller values of α. For sufficiently large ε, there is no need to explicitly construct G_{st}. Instead, localized techniques can be used to repeatedly solve flow and cut problems on small subgraphs until a global solution is reached [13, 16].

2.3 Related Spectral Methods Spectral methods are another widely popular approach to local graph clustering. Among these methods, the Andersen-Chung-Lang PUSH procedure for computing an approximate personalized PageRank vector is well-known for its strongly-local runtime and good cut improvement guarantees [1]. Random-walk based spectral methods typically find local cuts in a graph by running a localized diffusion from a small set of seed nodes. This diffusion produces an embedding with limited support over the nodes in the graph, which can then be rounded using some form of a sweet cut procedure to produce a cut. In contrast, flow-based methods solve biased minimum-cut computations and directly produce a cut rather than an embedding which must be rounded.

Another key distinction between random-walk and flow-based approaches is the type of seed set these methods require. Random-walk diffusions are typically able to grow a single seed node or a small seed set into a larger cluster with good conductance. Andersen et al. [1] showed that if one starts from any one of a large number of individual seed nodes in a target cluster T, the PUSH algorithm will return a localized cluster with conductance at most $O(\sqrt{\phi(T)})$. Flow-based methods are able to provide stronger cut improvement guarantees, but can only do so if they begin with a large seed set that has significant overlap with the target cluster T [2, 13, 16]. In practice, flow-based methods may perform poorly if the seed set R is too small. One approach for obtaining a large enough seed set is to first run a spectral diffusion from a small number of starting nodes and then refine the output using the flow-based method. Another approach is to take the starting seed nodes and grow them by a neighborhood with a small radius to produce a large input seed set.

3 Generalized Local Clustering Objective

In order to develop a flow-based method that places a higher emphasis on agreeing with the seed set, we begin by presenting a generalization of the local conductance objective (2.1). After introducing the objective, we prove cut improvement guarantees that can be achieved if this objective is minimized in practice.

3.1 The Seed-Penalized Conductance Score Let $G = (V, E)$ be undirected and unweighted, and R be a small set of nodes that we wish to grow into a larger cluster S. Unlike other methods, we assume there exists a designated set of nodes $R_s \subseteq R$ which must be included in the output set, and a weight $p_i \geq 0$ for every other node $r_i \in R$ which indicates our level of confidence that r_i should be included in S. We start by introducing a new *overlap score* between R and S:

$$\mathcal{O}_R(S) = \mathbf{vol}(R \cap S) - \varepsilon \mathbf{vol}(S \cap \bar{R}) - \textstyle\sum_{r \in R} p_r d_r \chi_{\bar{S}}(r)$$

where $\mathbf{p} = (p_i)$ is the vector of penalty weights for nodes in the seed set, $\chi_{\bar{S}}$ is the indicator function for nodes in $\bar{S}$, and ε is a locality parameter controlling how much we allow the output set to include nodes outside R. The first term rewards a high intersection between S and R, the second penalizes the inclusion of nodes outside R, and the third term introduces a penalty for nodes in R that are not in S. Our goal is to minimize the following *seed-penalized conductance* score:

$$\pi_R(S) = \begin{cases} \frac{\mathbf{cut}(S)}{\mathcal{O}_R(S)} & \text{if } \mathcal{O}_R(S) > 0, R_s \subseteq S \\ \infty & \text{otherwise.} \end{cases} \tag{3.2}$$

To keep notation simple, we only include the set R in the subscript of $\mathcal{O}_R(S)$ and $\pi_R(S)$, though we note that these also depend on R_s, ε, and $\mathbf{p}$, which are fixed parameters chosen at the outset of a problem.

3.2 Cut Improvement Guarantee Despite the differences between (3.2) and standard conductance, we can prove that minimizing the former will give strong cut improvement guarantees in terms of the latter. This result and its proof closely follow the cut improvement guarantees for local conductance variants shown in previous work [2, 16]. We update and alter these approaches to focus on the case where R is completely contained in some ground truth target cluster T, since in our work we are especially concerned with growing a seed set into a larger cluster.

THEOREM 3.1. *Let $G = (V, E)$ be connected and R be a seed set. Let T be any set of nodes containing R with $\boldsymbol{vol}(T) \leq \boldsymbol{vol}(\bar{T})$, and assume that $\boldsymbol{vol}(R) = \gamma \boldsymbol{vol}(T)$ for some $\gamma \in (0,1)$. For any $\varepsilon \in \left[\frac{2\boldsymbol{vol}(R)}{\boldsymbol{vol}(G) - 2\boldsymbol{vol}(R)}, \frac{\gamma}{1-\gamma}\right)$, if S^* is the set of nodes minimizing objective (3.2), then $\phi(S^*) \leq C\phi(T)$ where $C = \frac{1}{\gamma + \varepsilon\gamma - \varepsilon}$.*

Proof. For notational simplicity let $v_S = \mathbf{vol}(S)$ and $c_S = \mathbf{cut}(S)$ for any set S. Noting that $0 < \mathcal{O}_R(S^*) < v_{(R \cap S^*)} - \varepsilon v_{(\bar{R} \cap S^*)} \implies 0 < (1+\varepsilon) v_{(R \cap S^*)} - \varepsilon v_{S^*}$ allows us to bound the volume of S^*: $v_{S^*} < (1 + 1/\varepsilon) v_R$. Combining this bound with the lower bound on ε given in the theorem statement, we see that the volume of S^* is less than $\mathbf{vol}(G)/2$. Since $v_{S^*} > \mathcal{O}_R(S^*)$, we have

$$\phi(S^*) = c_{S^*}/v_{S^*} < c_{S^*}/\mathcal{O}_R(S^*) = \pi_R(S^*).$$

Because we assumed R is contained in T, we have $\sum_{r \in R} p_r d_r \chi_{\bar{T}}(r) = 0$ and $v_{(T \cap R)} = v_R$. Therefore,

$$\begin{aligned} \mathcal{O}_R(T) &= v_R - \varepsilon v_{(T \cap \bar{R})} = (1+\varepsilon) v_R - \varepsilon v_T \\ &= v_T((1+\varepsilon)\gamma - \varepsilon). \end{aligned}$$

Finally, since S^* minimizes (3.2),

$$\phi(S^*) < \pi_R(S^*) \le \pi_R(T) = \frac{c_T}{\mathcal{O}_R(T)}$$
$$= \frac{1}{(1+\varepsilon)\gamma - \varepsilon}\frac{c_T}{v_T} = C\phi(T). \qquad \blacksquare$$

If we select ε to be at its lower bound defined above, the approximation ratio will be $C = \frac{\mathbf{vol}(G)-2\mathbf{vol}(R)}{\gamma\mathbf{vol}(G)+\gamma\mathbf{vol}(R)-2\mathbf{vol}(R)}$. If $\mathbf{vol}(R)$ is very small compared to the overall size of the graph, then the approximation factor goes to $1/\gamma$ as the size of the graph increases for a fixed R.

3.3 Minimizing Seed-Penalized Conductance

As is the case for local conductance, objective (3.2) can be minimized in polynomial time by solving a sequence of minimum s-t cut problems. Fix $\alpha \in (0,1)$ and assume we wish to find whether there exists some S such that $\pi_R(S) < \alpha$. We construct a new version of the *cut graph* G_{st} which includes all nodes in G and an additional source s and sink t. For any node $r \in R_s$, we will assign a penalty variable $p_r = \mathbf{vol}(G)/\alpha$. We add an edge from s to each $r \in R$ with weight $\alpha(1+p_r)d_r$. The chosen weight for nodes in R_s is large enough to guarantee that a minimum cut will never separate any of these nodes from s. Then, for each node $w \in \bar{R}$, we add an edge from w to t with weight $\alpha\varepsilon d_w$.

For any set of non-terminal nodes $S \subseteq V$, the s-t cut associated with that set can be expressed in terms of cuts and volumes in the original graph G:

$$\mathbf{cut}(S) + \alpha\varepsilon\mathbf{vol}(\bar{R}\cap S) + \alpha \textstyle\sum_{r\in R} d_r(1+p_r)\chi_{\bar{S}}(r).$$

Observing that $\alpha\mathbf{vol}(R\cap\bar{S}) = \alpha\mathbf{vol}(R) - \alpha\mathbf{vol}(R\cap S)$, we rearrange this into the following objective function:

$$(3.3) \qquad f^{\alpha}_{R,\varepsilon}(S) = \mathbf{cut}(S) - \alpha\mathcal{O}_R(S) + \alpha\mathbf{vol}(R),$$

so $f_{R,\varepsilon}(S) < \alpha\mathbf{vol}(R)$ if and only if $\mathbf{cut}(S)/\mathcal{O}_R(S) < \alpha$. Thus, solving the minimum s-t cut objective on G_{st} will tell us whether there exists some S with seed-penalized conductance less than α. Given any procedure for minimizing objective (3.3) (e.g. a generic minimum s-t cut solver), we can minimize seed-penalized local conductance using Algorithm 1. We end this section by showing a bound on the number of iterations for Algorithm 1, by slightly adapting the techniques Andersen and Lang [2] used to prove a similar bound for a more restrictive objective function.

Theorem 3.2. *Algorithm 1 will need to solve min-cut objective* (3.3) *at most* $\mathbf{cut}(R)$ *times.*

Proof. Since R and ε and $\mathbf{p}$ are fixed at the outset of the algorithm we will write f^α instead of $f^\alpha_{R,\varepsilon}$ and $\mathcal{O}$ rather than $\mathcal{O}_R$. Consider two consecutive iterations in which Algorithm 1 successfully finds sets with improved seed-penalized conductance and therefore doesn't terminate. Let S_i be the set returned after the $(i-1)$st iteration, so $S_i = \arg\min f^{\alpha_{i-1}}(S)$ for some α_{i-1}, and set $\alpha_i = \pi_R(S_i) = \mathbf{cut}(S_i)/\mathcal{O}(S_i) < \alpha_{i-1}$. Similarly, $S_{i+1} = \arg\min f^{\alpha_i}(S)$ and $\alpha_{i+1} = \pi_R(S_{i+1}) < \alpha_i$. Note that

$$\begin{aligned} f^{\alpha_{i-1}}(S_i) &= \alpha_{i-1}\mathbf{vol}(R) + \mathbf{cut}(S_i) - \alpha_{i-1}\mathcal{O}(S_i) \\ &= \alpha_{i-1}\mathbf{vol}(R) + \mathcal{O}(S_i)(\pi_R(S_i) - \alpha_{i-1}) \\ &= \alpha_{i-1}\mathbf{vol}(R) + \mathcal{O}(S_i)(\alpha_i - \alpha_{i-1}) \end{aligned}$$

and $f^{\alpha_{i-1}}(S_{i+1}) = \alpha_{i-1}\mathbf{vol}(R) + \mathcal{O}(S_{i+1})(\alpha_{i+1} - \alpha_{i-1})$. Because S_i minimizes $f^{\alpha_{i-1}}$ we know that $f^{\alpha_{i-1}}(S_i) \le f^{\alpha_{i-1}}(S_{i+1})$, which implies that

$$\mathcal{O}(S_i)(\alpha_i - \alpha_{i-1}) \le \mathcal{O}(S_{i+1})(\alpha_{i+1} - \alpha_{i-1})$$

and since $(\alpha_{i+1} - \alpha_{i-1}) < (\alpha_i - \alpha_{i-1}) < 0$ we see that $\mathcal{O}(S_{i+1}) < \mathcal{O}(S_i)$. Thus both $\pi_R(S)$ and its denominator are strictly decreasing during the course of the algorithm, so $\mathbf{cut}(R)$ must strictly decrease at each step as well. Since we assume the graph is unweighted, there are at most $\mathbf{cut}(R)$ iterations in total. $\blacksquare$

Algorithm 1 Minimizing seed-penalized conductance

```
Input: $G$, $R$, $\varepsilon$, $\mathbf{p}$
Set $\alpha := 2$, $\alpha_{new} = \pi_R(R) = \phi(R)$, and $S = R$
while $\alpha_{new} < \alpha$ do
    $S_{best} \leftarrow S$
    $\alpha \leftarrow \alpha_{new}$
    $S \leftarrow \arg\min f^{\alpha}_{\varepsilon,R}(S)$
    $\alpha_{new} \leftarrow \pi_R(S)$
end while
Return: $S_{best}$
```

4 The Strongly-Local Meta-Procedure

The results of the previous section imply that Algorithm 1 can be run in polynomial time using any black-box min s-t cut solver. In this section we will prove a much stronger result by showing that objective (3.3) can be minimized in strongly-local time using a very simple two-step meta-procedure. A significant feature of this meta-procedure is that the algorithm does not require any explicit computation of maximum flows, but relies on a very simple repeated two-step procedure for localized minimum s-t cuts.

Local Graph Operations. In order minimize (3.3) without touching all of $G = (V,E)$, we will repeatedly solve a variant of objective (3.3) on a growing subgraph $L = (V, E_L)$ called the *local graph*, which contains a restricted edge set $E_L \subset E$. In theory L is assumed to have the same node set V but many of these nodes will have degree zero in L, so we will not need to explicitly perform computations with them in practice.

We consider the following localized variant of (3.3), which corresponds to a minimum s-t cut problem on a subgraph L_{st} of the cut graph G_{st}:

$$(4.4) \qquad f_L^\alpha(S) = \mathbf{cut}_L(S) - \alpha \mathcal{O}_R(S) + \alpha \mathbf{vol}(R).$$

The only difference from (3.3) is that $\mathbf{cut}_L(S)$ is defined to the be number of edges in E_L between S and $\bar{S}$, rather than the number of edges in E. Thus $f_L^\alpha(S) \leq f_G(S) = f_{R,\varepsilon}^\alpha(S)$ for all $S \subset V$ and for any such subgraph L of G. We will use the notation d_i^L to denote the degree of node i in L, which is always less than or equal to d_i. We then distinguish between an *edge-complete* set of nodes $L_C = \{i \in V : d_i = d_i^L\}$ and an *edge-incomplete* nodes $L_I = \{i \in V : d_i > d_i^L\}$ in L.

Let S_L be the minimizer of (4.4) for a fixed subgraph L. The following lemma shows that if S_L is made up entirely of edge-complete nodes, then this set also minimizes the global objective function $f_G = f_{R,\varepsilon}^\alpha$ (3.3).

LEMMA 4.1. *Let $S_L = \arg\min f_L(S)$. If $S_L \subseteq L_C$ then $S_L = \arg\min f_G(S)$.*

Proof. Because $E_L \subseteq E$, $\mathbf{cut}_L(S) \leq \mathbf{cut}(S)$ for all $S \subset V$, and therefore $f_L(S) \leq f_G(S)$ for all $S \subset V$, which implies that $\min_S f_L(S) \leq \min_S f_G(S)$. For the specified set S_L, since $S_L \subseteq L_C$, all nodes in S_L have the same degree in L as well as G, implying that $\mathbf{cut}_L(S_L) = \mathbf{cut}(S_L)$. Therefore:

$$f_L(S_L) = f_G(S_L) \geq \min_S f_G(S) \geq \min_S f_L(S) = f_L(S_L).$$

so equality holds throughout and S_L optimizes f_G. ■

Our meta-procedure for minimizing (3.3) in strongly local time operates by repeatedly solving objective (4.4) over a sequence of growing local subgraphs. This proceeds until an iteration in which the current subgraph L is large enough so that the set minimizing (4.4) is made up of edge-complete nodes, at which point we know by Lemma 4.1 that we have globally solved objective (3.3). The full procedure is given in Algorithm 2. The following result proves that the size of the largest subgraph formed by Algorithm 2 will be bounded in terms of $\mathbf{vol}(R)$, and ε. This mirrors a result for our previously developed meta-procedure [16], which required explicit computation of flows in order to solve an objective related to (3.3). The proof technique is very similar, though in order to avoid explicit computation of flows, more analysis is needed to prove the theoretical bound. We include a proof in the full version of the paper [17].

Algorithm 2 LOCAL MIN-CUT META-PROCEDURE

```
Input: graph $G$, seed set $R$, parameters $\alpha, \varepsilon$, $\mathbf{p}$
Initialize $L$: $L_C = R$, $L_I = \bar{R}$
$E_L$: all edges in $E$ with at least 1 endpoint in $R$.
repeat
  1. Solve Local Objective on $L$
  $S_L = \arg\min_S f_L^\alpha(S)$
  $N = S_L \cap L_I$ (new nodes to explore around)
  2. Expand $L$ around $N$
  for all $v \in N$ do
    $E_v$ = edges incident to node $v$ in $G$
    $E_L \leftarrow E_L \cup E_v$
    $L_C \leftarrow L_C \cup \{v\}$ and $L_I \leftarrow L_I - v$.
  end for
  $L \leftarrow (V, E_L)$
until $N = \emptyset$
```

THEOREM 4.1. *Let α be chosen so that $\pi_R(S_0) = \alpha$ for some $S_0 \subset V$. The volume of the largest subgraph L formed by Algorithm 2 is bounded above by $\textbf{vol}(L) \leq \textbf{vol}(R)\,(1 + 1/\varepsilon) + \textbf{cut}(R)$.*

The volume bound given here is the same as the bound shown for our previous method SIMPLELOCAL [16]. Thus, using Algorithm 2 as a subroutine in Algorithm 1 produces an algorithm with the same theoretical runtime as SIMPLELOCAL, despite solving a more general objective function and completely avoiding explicit maximum flow computations. Each flow problem takes at most $O(\mathbf{vol}(R)^3/\varepsilon)$ operations if fast flow subroutines are used [14], and Theorem 3.2 guarantees it will be run at most $\mathbf{cut}(R)$ times. This runtime bound is very conservative and the empirical performance will typically be significantly better.

5 The Push-Relabel Implementation

We implement Algorithm 2 using a new method for computing minimum s-t cuts based on a variant of the push-relabel algorithm of Goldberg and Tarjan [6]. The full push-relabel algorithm can be separated into two phases: the first phase computes a maximum preflow which can be used to solve the minimum s-t cut problem, and the second phase performs additional computations to turn the preflow into a maximum s-t flow. Because we only require minimum s-t cuts, our method simply applies Phase 1.

5.1 Push-Relabel Overview We give a basic overview of the maximum s-t flow problem in the full version of the paper [17]. The push-relabel algorithm is specifically a *preflow* algorithm for maximum flows, meaning that during the course of the algorithm, all arcs satisfy capacity constraints, but each node i is allowed to have more incoming flow than outgoing flow, i.e. a preflow satisfies a relaxation of the the flow constraints:

$$\sum_{(j,i)\in A} f_{ji} \leq \sum_{(i,k)\in A} f_{ik} \text{ for } i \in V$$

where $F = (f_{ij})$ is a flow assignment for a directed graph G with node set V and arc set A. Push-relabel maintains a labeling function $\ell : V \rightarrow \{0, 1, 2, \ldots, n\}$ where $n = |V|$ is the number of nodes in a graph G_{st} with distinguished source and sink. The algorithm can be initialized using any preflow and a labeling that gives a lower bound on the distance from each node to the sink in the residual graph. The standard initialization is to set $\ell(s) = n$ and the label of all other nodes to zero. The preflow is initialized to be zero on all edges, and afterwards all edges from s to its neighbors are saturated. This creates a positive *excess* at these neighbors, i.e. more flow goes into the nodes than out. After initialization, the algorithm repeatedly visits *active* nodes, which are nodes that have a label less than n and a positive excess. For a selected active node u, the algorithm locally pushes flow across admissible edges, which are defined to be edges (u, v) for which $\ell(u) = \ell(v) + 1$. If no admissible edges exist, the label of the node is increased to be the minimum label such that an admissible arc is created. During the course of the algorithm, it can be shown that $\ell(u) < \ell(v)$ for any arc (u, v) with nonzero residual capacity, and furthermore $\ell(v)$ is a lower bound on the distance from node v to the sink t, if there still exists a path of unsaturated edges from v to t. Phase 1 of the algorithm is complete when there are no more active nodes to process. At this point the preflow is at a maximum, and the set of nodes with label n forms the minimum cut set.

5.2 Label Selection Variants and Relabeling Heuristics The generic push-relabel algorithm simply requires one to push flow across admissible edges whenever there still exist active nodes. This procedure is guaranteed to converge to the solution to the minimum cut problem, but better runtimes can be obtained by more carefully selecting the order in which to process active nodes. One approach is the first-in-first-out (FIFO) method, which begins by pushing all initial active nodes into a queue, and adding new nodes to the queue as they become active. Another approach is to continually select the highest-labeled node at each step.

The push-relabel method can be made very fast in practice using efficient relabeling heuristics [4]. One simple but very effective heuristic is to periodically run a breadth first search from the sink node t and update the labels of each node to equal the distance from that node to t. Another heuristic is the gap relabeling heuristic, which checks whether there exist certain types of gaps in the labels that can be used to prove when certain nodes are no longer connected to the sink node t.

5.3 Implementation Details and Warm-Start Heuristic In practice we implement the FIFO push-relabel algorithm in the Julia programming language and make use of the global relabeling heuristic. Although implementations of push-relabel in other languages have made efficient use of the highest-label variant and the gap relabeling heuristic [4], these require slightly more sophisticated data structures that are more challenging to maintain in Julia. Our implementation choices make it possible to maintain a very simple but efficient method to implement Algorithm 2. Running this procedure for various α using Algorithm 1 provides a fast local graph clustering algorithm. Because our method is flow-based and puts a higher emphasis on including seed nodes, we refer to it as FlowSeed.

An important part of our implementation is a warm-start heuristic for computing consecutive minimum s-t cuts on the growing subgraph. Each local subgraph L corresponds to a local cut graph L_{st} with added source and sink nodes. For the first local cut graph, we use the standard initialization for push-relabel, i.e. start with a preflow of zero and saturate all edges from s to its neighbors. Applying push-relabel will return a maximum preflow F on L_{st}, and thus a minimum s-t cut which we use to update L as outlined in Section 4. After L and L_{st} are updated, the goal is to find an updated minimum cut. Note that F is no longer a maximum preflow on the updated L_{st}, but it will still be a valid preflow. Our warm-start procedure therefore initializes the next run of the push-relabel method with the preflow F, and sets the label of each node to be its distance to the sink in the corresponding residual graph. Initializing each consecutive maximum preflow computation in this way will be much more efficient than re-constructing L_{st} from L at each step and starting with a preflow of zero.

6 Experiments

We demonstrate the performance of FlowSeed in several community detection experiments and large scale 3D image segmentation problems. Code for our method and experiments is available online at `https://github.com/nveldt/FlowSeed`.

6.1 Community Detection Our first experiment demonstrates the robustness of FlowSeed in local community detection. We consider four graphs from the SNAP repository [10]: DBLP, Amazon, LiveJournal, and Orkut. Each network come with sets of nodes representing so-called "functional communities" [20], e.g.

user groups in a social network or product categories on Amazon. For each graph we select the ten largest communities, ranging in size from a few hundred to a few thousand nodes. We randomly select 5% of the nodes in each target community and grow these nodes by their immediate neighborhood to create a seed set R.

We compare several standard local graph clustering algorithms that come with strong locality guarantees: Push [1], HK-relax [7], CRD [18], SimpleLocal [16], and FlowSeed. For FlowSeed and SimpleLocal we use a locality parameter of $\varepsilon = 0.1$. We force FlowSeed to return sets containing the known 5% of the target community, but we don't include soft penalties on excluding other nodes. For HK-relax and Push we test a range of common parameter settings and return the set with best conductance, as is standard in practice [7]. For CRD we use the standard parameter settings recommended by the authors [18]. For HK-relax, Push, and CRD we also tried using just the known 5% of the target nodes as a seed set, but this was not effective in practice for any of the methods. Similarly, trying known target nodes one at a time as individual seed nodes and returning the best conductance output was also ineffective.

In Table 1, we report conductance, F1 scores, and runtimes for each method using the full seed set R, averaged over the 10 communities for each network. FlowSeed returns the best result among all methods on three of four datasets. The relative performance of the other four methods varies significantly depending on the dataset. Thus FlowSeed is able to provide more consistent and robust results thanks to its ability to better agree with semi-supervised information about the target communities. In the full version of the paper [17], we provide more information about the datasets, algorithm implementations used, and parameter settings. We also report more detailed results including average set size, precision, and recall for each method.

SimpleLocal and FlowSeed trade off in runtime for the above experiments, given that they are solving slightly different objectives. In order to provide a clearer runtime comparison between each method's underlying flow subroutine, we re-run both methods with a small locality parameter $\varepsilon = 5\mathbf{vol}(R)/\mathbf{vol}(\bar{R})$. For FlowSeed, we do not penalize the exclusion of seed nodes, which ensures that the two methods optimize the same exact objective. The average runtimes for the two methods are then given as follows:

	dblp	amazon	livejour.	orkut
FlowSeed	5.4	1.3	107.8	229.3
SimpleLocal	17.6	3.5	134.9	632.2

Thus, while HK-relax and Push are still faster local clustering algorithms, our push-relabel and warm start heuristics lead to improved running times specifically for flow-based methods, which often are able to provide the best community detection results.

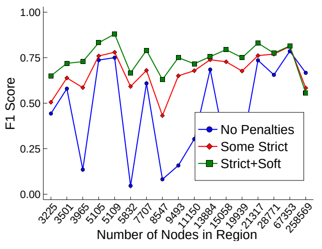
Table 1: Average conductance ϕ, F1 score, and runtime (in seconds) for five methods on four networks.

Graph		fl.seed	s.local	hk	push	crd
dblp	ϕ	0.25	0.05	0.10	0.13	0.26
	F1	**0.35**	0.01	0.04	0.02	0.26
	run	9.5	24.5	0.1	0.2	3.6
amazon	ϕ	0.02	0.01	0.01	0.01	0.21
	F1	**0.92**	0.81	0.84	0.90	0.52
	run	0.33	0.10	0.02	0.22	1.63
livejour.	ϕ	0.07	0.04	0.14	0.36	0.10
	F1	0.48	0.44	0.41	0.49	**0.52**
	run	22.8	17.9	0.21	0.17	69.6
orkut	ϕ	0.38	0.34	0.65	0.75	0.36
	F1	**0.49**	0.44	0.08	0.27	0.43
	run	439	327	3.8	0.77	451

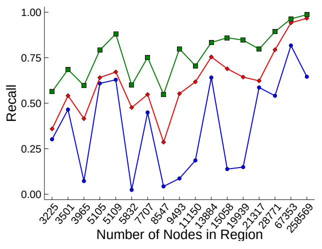
6.2 3D Image Segmentation on a Brain Scan

Next we turn to detecting target regions in a large graph constructed from a brain MRI. The data is made up of a labeled $256 \times 287 \times 256$ MRI obtained from the MICCAI-2012 challenge [12]. In previous work [16] we demonstrated how to convert the image into a nearest neighbors graph on the 3D voxels, using an approach similar to Shi and Malik [15]. The resulting graph has 18 million nodes, nearly 234 million undirected edges, and contains 95 ground truth labeled regions of the brain (e.g. ventricles, amygdalas, brain stem, etc.), ranging from 3104 to over 250,000 nodes in size, and with (weighted) conductance scores between 0.04 and 0.25.

Benefit of Seed Exclusion Penalties. We split up the 95 regions into a set of 17 example regions spanning a broad range of cluster sizes. We test a range of parameters on these example regions to observe how different locality parameters and seed exclusion penalties behave for different sized regions and seed sets. We test locality parameters $\varepsilon \in \{0.5, 0.25, 0.1, 0.05\}$ and construct seed sets by taking very small subsets of the target cluster and growing them by their neighborhood. We find that with almost no exceptions, including strict and soft seed exclusion penalties leads to significant benefits in ground truth recovery across all region sizes. In Figure 1, for the 17 example regions we plot the recall and F1 scores for region recovery for a locality parameter of $\varepsilon = 0.1$ in the case where the seed set

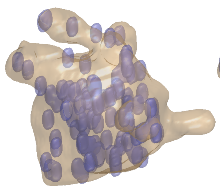


(a) F1 score

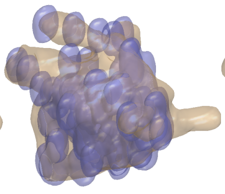


(b) Recall

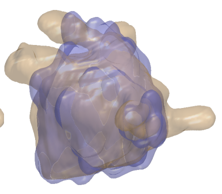
Figure 1: When detecting target regions of a large brain graph, enforcing no penalties on excluding seed nodes (blue) allows the method to discard too many seeds, often leading to very poor recall. If we enforce strict penalties on excluding known target nodes (red), this significantly improves recall and hence overall detection of the target cluster in terms of F1 score. We see even greater improvement by including a soft penalty for excluding any other node in the seed set (green).



(a) PUSH $\varepsilon_{pr} = 10^{-8}$



(b) PUSH $\varepsilon_{pr} = 10^{-10}$



(c) FLOWSEED

Figure 2: The yellow region indicates the target atrial cavity in a full-body MRI scan and purple regions represent algorithm output. Starting from a seed set of 100 target nodes plus neighbors, PUSH forms disconnected circular regions, which grow as tolerance parameter ε_{pr} decreases. We refine the $\varepsilon_{pr} = 10^{-10}$ region with FLOWSEED to produce a connected region which does a much better job identifying the region boundary and has significantly better F1 score (c).

is made up of a random sample of 2% of the target region plus neighbors. We run our method with (1) no penalties on excluding seed nodes, (2) strict penalties on excluding the initial 2% of nodes, and (3) strict penalties on the 2% and a soft penalty of $p_r = 1$ for excluding any other seed node. As expected, when we include no penalties, the flow-based approach often shrinks the seed set into a small cluster with good conductance and precision, but almost no recall. As we increase the strength of seed exclusion penalties, the precision decreases slightly but the recall improves considerably, leading to a much better overall ground truth recovery.

Comparison with Random-Walk Methods We additionally run PUSH with PageRank (pr) teleportation parameters α_{pr} from 0.5 to 0.9, and approximation tolerance parameters ε_{pr} from 10^{-11} to 10^{-7}. This ε_{pr} controls how wide a region is explored in the graph, much like FLOWSEED's locality parameter ε.

For both PUSH and FLOWSEED, we use observations from the experiments on the 17 example regions to inform our choice of parameter settings for different sized regions and seed set sizes. We then use these parameters to test the performance of each method on the remaining 78 regions, which we refer to as the evaluation set. We run experiments for the case where we know exactly 100 of the target nodes, and where 1%, 2%, and 3% of the target region is given. We run PUSH with a teleportation parameter of $\alpha_{pr} = 0.6$, and run FLOWSEED with both strict and soft penalties. In each experiment we identify which of the 17 example regions is closest in size to the target region from the evaluation set, and then set ε and ε_{pr} to be the values which led to the best F1-score recovery for this comparable example region.

Our experiments highlight a tradeoff in the performance of the two algorithms. For small seed sets, PUSH typically outperforms FLOWSEED in ground truth recovery, as it is able to grow a very small seed set into a sizable cluster. However, given sufficient information regarding the target cluster, we see a distinct benefit in applying our flow-based approach. When 2% (resp. 3%) of the target is known, FLOWSEED obtains a higher F1 score for 64 (resp. 68) of the 78 target regions, and the scores are on average 6.2% (resp. 6.1%) higher than those returned by PUSH. In terms of runtime, the highly optimized PUSH implementation is faster: most experiments run in under 1 second, with the largest taking several seconds. Our method takes up to 15 minutes for the largest region, but typically runs in 10-60 seconds for small and medium sized regions.

6.3 Detecting an Atrial Cavity

In our last experiment we demonstrate that random-walk and flow-based methods can be viewed as complementary approaches rather than competing algorithms. We combine the strengths of PUSH and FLOWSEED to provide good quality 3D segmentations of a manually labeled left atrial cavity in a whole-body MRI scan, provided as a part of the 2018 Atrial Segmentation Challenge [19]. We convert one such MRI into a graph with 29.2 million nodes and 390 million edges. The cavity in the MRI corresponds to a target cluster with 252,364 nodes and a conductance of 0.0414 in the graph.

We begin from a small set of 100 randomly selected nodes from the atrial cavity, constituting less than 0.04% of the target region. We grow these nodes by their neighborhood and use this as input to the PUSH algorithm. We again set $\alpha_{pr} = 0.6$ and test a range of tolerance parameters ε_{pr} from 10^{-14} to 10^{-8}. The best F1 score achieved is 0.714 (precision is 0.666, recall is 0.768) when $\varepsilon_{pr} = 10^{-12}$. In Figure 2 we show the output of the PUSH algorithm for $\varepsilon_{pr} = 10^{-8}$ and $\varepsilon_{pr} = 10^{-10}$, which indicate that the method is simply returning ball-shaped regions around the initial seeds, with an increasing radius as ε_{pr} decreases. Many of the output sets are not connected. We then take the output of PUSH and refine it using FLOWSEED with locality parameter $\varepsilon = 0.1$. We set a strict penalty on excluding the original 100 nodes from the cavity, a soft penalty of 1 on excluding their neighbors, and a penalty of 0.5 on excluding any node in the set returned by PUSH. We see a significant improvement in quality of segmentation, in the best case leading to a precision of 0.8498, recall of 0.7571, and F1 score of 0.8008 when refining the region output by PUSH when $\varepsilon_{pr} = 10^{-10}$. In Figure 2c we see that FLOWSEED smooths out the circular regions returned by PUSH to return a connected region that better identifies the boundary of the target cavity. Regarding runtime, PUSH quickly grows circular regions within a few seconds, and the FLOWSEED refinement procedure takes just under an hour. Together these methods produce a significantly better output than either method could have accomplished alone.

7 Conclusions and Future Work

In our work we have exploited efficient warm-start and push-relabel heuristics to provide practitioners with a very simple yet extremely fast flow-based method for local graph clustering. Our method is additionally able to obtain more robust results in community detection and large scale image segmentation experiments by giving users the option to specify penalties and strict constraints for excluding specific seed nodes from the output set. Given the success of seed exclusion penalties for flow-based methods, in future work we will continue to explore how similar penalties may be incorporated in other well-known clustering approaches including spectral and random-walk based techniques.

References

[1] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using PageRank vectors. In *FOCS 2006*, 2006.

[2] Reid Andersen and Kevin Lang. An algorithm for improving graph partitions. In *SODA 2008*, pages 651–660, January 2008.

[3] Reid Andersen and Kevin J. Lang. Communities from seed sets. In *WWW 2006*, pages 223–232, 2006.

[4] B. V. Cherkassky and A. V. Goldberg. On implementing the push—relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, Dec 1997.

[5] Yefim Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Doklady Akademii nauk SSSR*, 11:1277–1280, 1970.

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

[7] Kyle Kloster and David F. Gleich. Heat kernel based community detection. In *KDD 2014*, pages 1386–1395, 2014.

[8] Isabel M. Kloumann and Jon M. Kleinberg. Community membership identification from small seed sets. In *KDD 2014*, pages 1366–1375, 2014.

[9] Kevin Lang and Satish Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *IPCO 2004*, pages 325–337, 2004.

[10] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[11] Michael W. Mahoney, Lorenzo Orecchia, and Nisheeth K. Vishnoi. A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally. *J. Mach. Learn. Res.*, 13(1):2339–2365, August 2012.

[12] Daniel S. Marcus, Tracy H. Wang, Jamie Parker, John G. Csernansky, John C. Morris, and Randy L. Buckner. Open access series of imaging studies (oasis): Cross-sectional mri data in young, middle aged, nondemented, and demented older adults. *J. Cognitive Neuroscience*, 19(9):1498–1507, 2007.

[13] Lorenzo Orecchia and Zeyuan Allen Zhu. Flow-based algorithms for local graph clustering. In *SODA 2014*, pages 1267–1286, 2014.

[14] James B. Orlin. Max flows in o(nm) time, or better. In *STOC 2013*, pages 765–774, 2013.

[15] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI 2000*, pages 888–905, August 2000.

[16] Nate Veldt, David Gleich, and Michael Mahoney. A simple and strongly-local flow-based method for cut improvement. In *ICML 2016*, volume 48, pages 1938–1947, New York, New York, USA, June 2016.

[17] Nate Veldt, Christine Klymko, and David F. Gleich. Flow-based local graph clustering with better seed set inclusion. *arXiv preprint arXiv:1811.12280*, 2019.

[18] Di Wang, Kimon Fountoulakis, Monika Henzinger, Michael W. Mahoney, and Satish Rao. Capacity releasing diffusion for speed and locality. In *ICML 2017*, pages 3598–3607. PMLR, 06–11 Aug 2017.

[19] Zhaohan Xiong, Vadim V Fedorov, Xiaohang Fu, Elizabeth Cheng, Rob Macleod, and Jichao Zhao. Fully automatic left atrium segmentation from late gadolinium enhanced magnetic resonance imaging using a dual fully convolutional neural network. *IEEE Transactions on Medical Imaging*, 2018.

[20] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, Jan 2015.