A Simple and Strongly-Local Flow-Based Method for Cut Improvement

Nate Veldt LVELDT@PURDUE.EDU
Mathematics Department, Purdue University, West Lafayette, IN 47906

David F. Gleich DGLEICH@PURDUE.EDU
Computer Science, Purdue University, West Lafayette, IN 47906

Michael W. Mahoney MMAHONEY@STAT.BERKELEY.EDU
International Computer Science Institute and Dept. of Statistics, University of California at Berkeley, Berkeley, CA 94720

Abstract

Many graph-based learning problems can be cast as finding a good set of vertices nearby a seed set, and a powerful methodology for these problems is based on maximum flows. We introduce and analyze a new method for locally-biased graph-based learning called SimpleLocal, which finds good conductance cuts near a set of seed vertices. An important feature of our algorithm is that it is strongly-local, meaning it does not need to explore the entire graph to find cuts that are locally optimal. This method solves the same objective as existing strongly-local flow-based methods, but it enables a simple implementation. We also show how it achieves localization through an implicit ℓ_1-norm penalty term. As a flow-based method, our algorithm exhibits several advantages in terms of cut optimality and accurate identification of target regions in a graph. We demonstrate the power of SimpleLocal by solving problems on a 467 million edge graph based on an MRI scan.

1. Introduction and Related Work

Finding good conductance cuts near a set of seed vertices in a graph is a well-studied and widely-applied problem in graph-based learning. Such an algorithm is strongly-local if its runtime depends only on the size of the seed set or on the size of the output set rather than on the size of the entire graph. Seeded PageRank and other spectral and random-walk methods give these guarantees. While these methods provide nice approximation guarantees, they fail to give optimal solutions and often exhibit "sloppy" boundaries when it comes to solving label propagation and community detection problems.

Flow-based methods exhibit numerous advantages including the ability to provide exact optimal solutions in some cases. The first strongly local flow-based method was introduced by Orecchia & Zhu (2014), which exhibits a fast runtime but relies on a complicated variation of Dinic's algorithm for maximum flows, making it difficult to use in practice. In this paper we provide a new strongly-local algorithm that provides the same optimality guarantees while offering the flexibility of employing *any* max-flow algorithm as a subroutine. While our algorithm's theoretical runtime is weaker than Orecchia & Zhu (2014), we provide implementation details for it and demonstrate its ability to find low-conductance cuts in a large real-world dataset.

Graph-based learning. Graph-based learning is a recurring problem in machine learning where we are given a graph and some information about the nodes of this graph and the task is to infer the information on the unlabeled nodes. This is an instance of semi-supervised learning (Blum & Chawla, 2001; Zhu et al., 2003) or transductive learning (Joachims, 2003). Algorithms for these problems on graphs are often called *label propagation* methods due to their interpretation as spreading labels around a graph (Zhu et al., 2003; Fujiwara & Irie, 2014). Related problems include guided image segmentation (Mahoney et al., 2012) and seeded community detection (Andersen & Lang, 2006; Kloumann & Kleinberg, 2014), where we are given a set of sample pixels or nodes and the goal is to find the rest.

Algorithms for graph-based learning largely split into three types: flow-based methods, spectral methods, and graph-based heuristics. Some of the seminal papers in semi-

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

supervised learning on graphs and community detection discussed using minimum cuts in the network for this application (Blum & Chawla, 2001; Flake et al., 2000). Subsequent papers found that *spectral methods* had a number of advantages in terms of speed, unique solutions, and additional information about the *strength* of the prediction (Zhu et al., 2003; Joachims, 2003; Zhou et al., 2003). Principled heuristic methods also abound (e.g. Fujiwara & Irie 2014) due to the simplicity of the setup.

Semi-supervised learning algorithms differ from typical graph algorithms in that they exhibit special locality properties. Typical graph algorithms, for example Kruskal's minimum spanning tree method, often depend on optimizing an objective function over the entire graph structure and return a result proportional to the size of the entire graph. In contrast, semi-supervised learning algorithms take as input an exogenously specified seed set of nodes, and return results that are biased towards a small part of the graph nearby the seed or reference set. If the running time is still dependent on the entire size of the graph, this is called *weak locality.* For instance, solving the linear system involved in Zhou et al. (2003) returns a modestly sized set of nodes where the labels are expected to be located, but the linear system involves the entire graph.

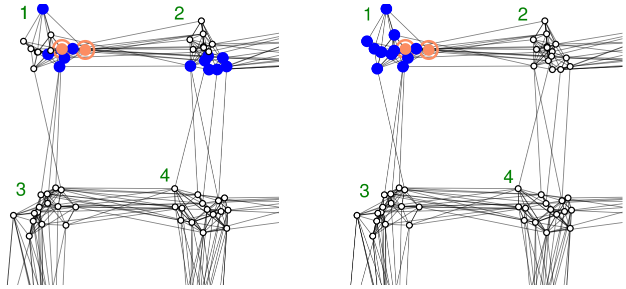
For spectral methods, Spielman & Teng (2013) and Andersen et al. (2006) have shown much stronger results. These algorithms are *strongly-local*, in that the algorithm doesn't even access most of the nodes of the graph and thus the running time is dependent on the size of the seed set or output set, rather than the entire graph. Importantly, not only are these strongly-local spectral algorithms very fast (in both theory and practice, see Leskovec et al. 2009 and Jeub et al. 2015) but, when interpreted as graph partitioning methods, they come with locally-biased Cheeger-like quality of approximation guarantees with respect to the conductance objective. (Good conductance means small conductance; we define conductance later.)

In comparison, flow-based methods can be shown to *optimally* solve the discrete partitioning objective, such as minimum conductance cut, if they are given an input that is not too large and that contains the desired set (Lang & Rao, 2004) through a parameteric flow construction (Gallo et al., 1989). A subsequent flow-based algorithm called Improve had related exactness guarantees on the discrete objective but considerably relaxed the requirements on the input set (Andersen & Lang, 2008). In the context of semi-supervised learning, the Improve algorithm is a weakly-local method that would essentially find the optimal conductance set of nodes in the graph that is nearby the set of the seed labels. (We make a precise statement in Section 3.1.) In fact, Improve is essentially a flow-based analog of the spectral method used by Zhou et al. (2003) as shown by Gleich & Mahoney (2015) using ideas from Mahoney et al. (2012). Recently, Orecchia & Zhu (2014) proposed a method called LocalImprove that combined a slight modification to the discrete objective function in the Improve algorithm with a variation on Dinic's algorithm for max-flow (Dinitz, 1970) in order to assemble the first flow-based method that is strongly-local.

Strengths and weaknesses with spectral methods for graph-based learning and some fixes. In recent work, spectral methods have been found to have *substantially* better theoretical guarantees, more akin to the guarantees of flow-based methods, if the resulting set of labels is *very well-connected internally* (Zhu et al., 2013). However, when that doesn't hold, spectral methods tend to produce *sloppy boundaries* unless the boundaries between labels is extremely clear. Figure 1 illustrates this effect on a simple synthetic construction. Given initial identical-labeled nodes, the spectral method (Zhou et al., 2003) diffuses over the boundary between groups too quickly and results in a potential misclassification. In contrast, the flow-based method (SimpleLocal) is able to correctly identify the boundary given the same seeds.

This example reflects a simplified scenario without the variety of fixes that are commonly used in spectral-based semi-supervised learning (Joachims, 2003; Zhou & Srebro, 2011; Lu & Peng, 2012; Brindle & Zhu, 2013). However, these spectral and strongly-local spectral methods have complicated theory with many parameters and options. This can make it difficult for non-experts to use, and it can be difficult to know what those fixes and strongly-local approximations are optimizing exactly.

Cut improvement. Cut improvement is a problem framework where we are given an initial partitioning of a graph into two pieces and the goal is to identify a *better* split according to some quotient of cut and size. Both conductance and its relative, *quotient cut*, fit into this general framework. Algorithms here date back to initial work on parametric maximum flow (Gallo et al., 1989). There are a variety of methods that use the submodular property of the objective and various decompositions of submodular functions to solve them (Patkar & Narayanan, 2003; Narasimhan & Bilmes, 2007). The original flow-based methods MQI and Improve had attractive theoretical guarantees and empirical performance for this task (Lang & Rao, 2004; Andersen & Lang, 2008). The tendency of spectral methods to make errors around the boundary was also recognized in this literature (Lang, 2005) and these flow-based methods were already a well-known fix. Along these lines, Chung (2007) provided a spectral analogue of MQI and Mahoney et al. (2012) provided MOV, a weakly-local version of spectral graph partitioning, which may be



(a) Graph-based learning with a spectral method

(b) Graph-based learning with a local flow method

Figure 1. There are four synthetic labels in displayed region of the graph, one for each group. The vertices are connected based on nearest neighbors and sparsified longer range connections. Using the two seed nodes marked in orange, the spectral prediction, the blue nodes in (a), move to the adjacent group whereas the flow predictions, the blue nodes in (b), accurately capture the true boundary.

interpreted as a spectral analogue of Improve. In fact, Gleich & Mahoney (2015) showed that the MOV method is optimizing the same solution as the semi-supervised method of Zhou et al. (2003).

Summary of Contributions. In this paper, we first show that the modification to the Improve objective used in the strongly-local LocalImprove can be precisely stated as an ℓ_1-penalized Improve objective (Theorem 1). This makes precise the sense in which LocalImprove *implicitly* optimizes a sparsity-induced regularized version of Improve. The authors of LocalImprove made use of a sophisticated white-box modification of Dinic's algorithm to provide the best possible runtime. Our main contribution is then a new strongly-local flow algorithm that uses existing max-flow algorithms as a black box (Algorithm 1, Theorem 2). Our algorithm solves the same optimization problem as LocalImprove and our aim is to provide an algorithm that is both flexible and easy to implement while still being strongly-local. Thus we call our algorithm SimpleLocal.

In combination, these two results deconvolve the origin of the local solution, which occurs because of the the ℓ_1-regularization applied to the problem, from the algorithm that identifies this local solution. The second result enables us to create an extremely simple and scalable implementation of SimpleLocal where *any* max-flow algorithm can be used to solve this sequence of problems including efficient GPU methods (He & Hong, 2010) or the latest exact theoretical algorithms (Orlin, 2013). We use this implementation to show a few examples of how this new method is able to solve problems on graphs originating from MRI data with 467 million edges in a few minutes.

2. Preliminaries and Notation

Let $G = (V, E)$ be an undirected, unweighted graph with $n = |V|$ nodes and $m = |E|$ edges. For a given vertex $v \in V$, the degree d_v is equal to the sum of edges incident to v, and the volume of a subset of nodes $S \subset V$ is defined to be $\text{vol}(S) = \sum_{s \in S} d_s$. Given two subsets of vertices A and B, we indicate the set of edges between them by

$$\text{cut}(A, B) = \text{cut}(B, A) = \{(i, j) \in E : i \in A, j \in B\}.$$

We associate every vertex set $S \subset V$ with the set of edges between S and the rest of the graph, $\text{cut}(S) = \text{cut}(S, \bar{S})$, where $\bar{S} = V \backslash S$. We use $\partial S = |\text{cut}(S, \bar{S})|$ to indicate the number of edges in this set. Let $\text{Neigh}(S)$ indicate the nodes that are not included in S but share an edge with S,

$$\text{Neigh}(S) = \{v \in \bar{S} : (v, s) \in E \text{ for some } s \in S\}.$$

We measure how well-connected the set S is by its conductance $\phi(S)$, defined by

$$\phi(S) = \frac{\partial S}{\min\{\text{vol}(S), \text{vol}(\bar{S})\}}.$$

3. Implicit Sparsity Regularization

In this section we present a new result which relates the objectives of Improve and LocalImprove. We begin by reviewing the construction of the *augmented graph* of Andersen & Lang (2008) used in Improve, and the modification of this graph introduced by Orecchia & Zhu (2014). We relate both of these graphs back to our original problem by showing how low-capacity cuts in the augmented and modified augmented graphs correspond to low-conductance cuts in the original input graph. Our main result in this section is to show that the min-cut objective solved by LocalImprove is implicitly equivalent to a sparsity-inducing ℓ_1-regularization of the min-cut objective solved by Improve. This result guides our understanding of strongly-localized flow-based cut improvement methods, and sheds light on the success and robustness of algorithms such as LocalImprove and SimpleLocal. This is also an example of algorithmic anti-differentiation (Gleich & Mahoney, 2014) where we characterize the optimization problem that LocalImprove was implicitly solving as a result of their algorithmic setup.

3.1. Improve and the Augmented Graph

We begin with a graph $G = (V, E)$, an initial seed set $R \subset V$ satisfying $\text{vol}(R) \leq \text{vol}(\bar{R})$, and a parameter $\alpha \in (0, 1)$. The *augmented graph* $G_R(\alpha)$ is constructed through the following steps:

1. Retain original nodes, edges, and edge weights of G
2. Add a source node s and a sink node t

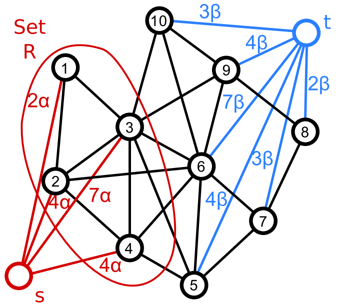


Figure 2. An illustration of the augmented graph $G_R(\alpha)$ used by Improve, where $\beta = \alpha f(R)$. If we change the sink-side weight so that $\beta = \alpha(f(R) + \delta)$, this corresponds to the modified augmented graph $G'_R(\alpha, \delta)$ used by LocalImprove and our SimpleLocal.

3. For every $r \in R$, add an edge (s, r) with capacity αd_r
4. For every $v \in \bar{R}$, add edge (v, t) with capacity $\alpha f(R) d_v$.

Here $f(R) = \frac{\text{vol}(R)}{\text{vol}(\bar{R})}$ is chosen so that the total capacity into the sink equals the total capacity out of the source. See Figure 2 for a visualization of a small augmented graph.

An s-t cut of the augmented graph is any set of edges which, when cut, partitions the nodes into disjoint sets where the source and sink are in separate sets. Any s-t cut can be associated with the set of nodes $S \subset V$ from the original graph that are on the same side of the cut as s. The capacity of any s-t cut of $G_R(\alpha)$ is equal to

$$\alpha \,\text{vol}(R) + \partial S - \alpha \,\text{vol}(R \cap S) + \alpha f(R) \,\text{vol}(\bar{R} \cap S), \qquad (1)$$

where S is the set of edges on the source side.

The Improve algorithm solves a sequence of minimum s-t cut problems on $G_R(\alpha)$ for decreasing values of α. This is exactly equivalent to solving a sequence of optimization problems where we minimize the objective (1) over sets $S \subset V$. The goal is to find the smallest α such that the minimum s-t cut (i.e. the optimal S) is less than $\alpha \,\text{vol}(R)$. Note that regardless of α we can always achieve a cut with capacity $\alpha \,\text{vol}(R)$ by selecting all edges from the source to the rest of the graph. This type of problem is an instance of a parametric max-flow (Gallo et al., 1989); although those techniques are unnecessary for Improve.

3.2. LocalImprove and the Modified Augmented Graph

The *modified augmented graph* $G'_R(\alpha, \delta)$ is obtained by increasing the weight of the edges from $\bar{R}$ to t to $\alpha \varepsilon d_w$ for all nodes $w \in \bar{R}$, where $\varepsilon = f(R) + \delta$ for $\delta \geq 0$. Finding the minimum s-t cut of $G'_R(\alpha, \delta)$ is equivalent to minimizing a slightly modified objective:

$$\begin{aligned} &\alpha \,\text{vol}(R) + \partial S - \alpha \,\text{vol}(R \cap S) + \alpha f(R) \,\text{vol}(\bar{R} \cap S) \\ &\qquad + \alpha \delta \,\text{vol}(\bar{R} \cap S). \end{aligned} \qquad (2)$$

By including the extra term $\alpha \delta \,\text{vol}(\bar{R} \cap S)$, this in effect increases the penalty of including nodes outside of R. Note that $G_R(\alpha) = G'_R(\alpha, 0)$.

Cuts in both $G_R(\alpha)$ and $G'_R(\alpha, \delta)$ are easily associated with sets of vertices in the original graph G. Given a max s-t flow on either graph, the set of nodes reachable from s via unsaturated edges (excluding s itself) forms a subset $S \subset V$ in the original graph G.

LocalImprove finds a set S with low conductance by solving a sequence of *approximate* max flow computations on $G'_R(\alpha, \delta)$ for different values of α. The method relies on modifying Dinic's max-flow algorithm and a procedure for finding blocking flows on a subgraph of $G'_R(\alpha, \delta)$ called the *local graph*. The local graph is updated and expanded as needed at each step to allow more flow to be routed from s to t. For more details we direct the reader to Orecchia & Zhu (2014).

3.3. Relating Cuts in Modified Augmented Graph to Low-Conductance Sets in the Original Graph

The following lemma is a generalization of a core result of Andersen & Lang (2008) and is proven as a part of Lemma 3.1 in Orecchia & Zhu (2014).

Lemma 1 *If the minimum s-t cut of $G'_R(\alpha, \delta)$ for $\delta \geq 0$ is less than $\alpha \,\text{vol}(R)$, then $\phi(S) < \alpha$, where S is the node set corresponding to the cut.*

(For a proof see the supplementary material.)

The task of Improve is to find the smallest α such that the maximum s-t flow of the augmented graph is less than $\alpha \,\text{vol}(R)$. This is equivalent to minimizing the function

$$\phi_R(S) = \partial S / (\text{vol}(R \cap S) - f(R) \,\text{vol}(\bar{R} \cap S)) \qquad (3)$$

which Andersen & Lang (2008) refer to as the *quotient score of S relative to the seed set R*. Both LocalImprove and our new algorithm SimpleLocal minimize a similar quotient function, given later as equation (7).

3.4. Relating s-t Cuts of the Augmented Graph and Modified Augmented Graph

Our first new result is to show that finding a minimum s-t cut of the modified augmented graph is equivalent to solving an ℓ_1 sparsity-regularized version of the min-cut objective for a related, standard augmented graph. This result gives insight into why LocalImprove succeeds in finding a cut with similar conductance guarantees to those provided by Improve, despite only exploring a portion of the graph. This result is analogous to a similar relationship discovered in Gleich & Mahoney (2014) between the Andersen, Chung, Lang procedure and an ℓ_1-regularized ℓ_2-version of the min-cut objective on the Improve augmented graph (Andersen et al., 2006).

Theorem 1 *Finding the minimum cut of the modified augmented graph $G'_R(\alpha,\delta)$ is equivalent to solving an ℓ_1-regularized version of the minimum s-t cut objective of a related augmented graph $G_R(\beta)$. More specifically, the set S which minimizes the s-t cut objective of $G'_R(\alpha,\delta)$:*

$$\alpha\,\mathrm{vol}(R) + \partial S - \alpha\,\mathrm{vol}(R \cap S) + (\alpha f(R) + \alpha\delta)\,\mathrm{vol}(\bar{R} \cap S) \quad (4)$$

also minimizes the ℓ_1-regularized objective of $G_R(\beta)$:

$$\beta\,\mathrm{vol}(R) + \partial S - \beta\,\mathrm{vol}(R \cap S) + \beta f(R)\,\mathrm{vol}(\bar{R} \cap S) + \kappa\,\mathrm{vol}(S), \quad (5)$$

where $\kappa = \frac{\alpha\delta}{1+f(R)} > 0$ and $\beta = \alpha + \kappa$.

Proof If we note that $\mathrm{vol}(\bar{R} \cap S) + \mathrm{vol}(R \cap S) = \mathrm{vol}(S)$ and substitute $\alpha\delta = \kappa + \kappa f(R)$ and $\alpha = \beta - \kappa$, we get

$$\begin{aligned} &-\alpha\,\mathrm{vol}(R \cap S) + (\alpha f(R) + \alpha\delta)\,\mathrm{vol}(\bar{R} \cap S) \\ &\quad = -(\beta - \kappa)\,\mathrm{vol}(R \cap S) + (\beta f(R) + \kappa)\,\mathrm{vol}(\bar{R} \cap S) \\ &\quad = -\beta\,\mathrm{vol}(R \cap S) + \beta f(R)\,\mathrm{vol}(\bar{R} \cap S) + \kappa\,\mathrm{vol}(S). \end{aligned}$$

The constant term in each objective does not affect the optimal set S, so we see that (4) and (5) are equivalent. ■

Remark. The additional term in the regularized objective (5) is $\kappa\,\mathrm{vol}(S)$. When this objective is converted into a linear program for min-cut, it is:

$$\begin{array}{ll} \min_x & \sum_{(i,j)\in E_R(\beta)} c_{i,j}(\beta)|x_i - x_j| + \kappa \sum_i |d_i x_i| \\ \text{s.t.} & 0 \le x_i \le 1, x_s = 1, x_t = 0, \end{array} \quad (6)$$

where $c_{i,j}(\beta)$ is the capacity of each edge in the augmented graph with β from the theorem and where d_i is the degree of node i in the original graph. The extra term is then exactly an ℓ_1 penalty.

4. SimpleLocal Algorithm

Our primary contribution is the algorithm SimpleLocal, a simplified framework for computing the objective of LocalImprove. Just as LocalImprove, we rely on constructing and updating a local subgraph of $G'_R(\alpha,\delta)$. However, rather than using Dinic's algorithm to compute approximate maximum flows, we develop a new three-stage method for exact maximum flow computations on $G'_R(\alpha,\delta)$.

4.1. Three-Stage Local Max Flow Procedure

We begin with a detailed explanation of 3StageFlow, the newly designed algorithm we employ to compute a maximum s-t flow of a given modified augmented graph $G'_R(\alpha,\delta)$. After constructing an initial local graph, our algorithm enters a three-stage process that is repeated until convergence to a maximum flow. In each iteration we expand the local graph, compute a small-scale maximum s-t flow, and then update the local graph based on this flow. By iteratively growing the local graph and increasing our small-scale flow computations on it in this way, we will converge to an s-t flow that is a maximum on all of $G'_R(\alpha,\delta)$.

Initialization. Let G' denote the modified augmented graph $G'_R(\alpha,\delta)$. We begin by forming the local graph $L = (V_L, E_L)$, a subgraph of G' which includes:

- Nodes $\{s,t\} \cup R \cup \mathrm{Neigh}(R)$
- Edges from s to R
- Edges between distinct nodes in R
- Edges from R to $\mathrm{Neigh}(R)$
- Edges from t to $\mathrm{Neigh}(R)$.

Let F denote our flow vector, and *flow*(F) indicate the total amount of flow routed from s to t. Initially F is set to the zero vector.

Stage 1. Expansion. At the beginning of each new iteration we expand the local graph to allow more flow to be routed from s to t. We use X to denote the set of nodes to expand on at the beginning of an iteration. For any node $x \in X$, we add all neighbors of $x \in G'$ that are not yet a part of L, and also include all edges from x to all its neighbors. For each new node added to L, we include the edge it shares with the sink t. In the first step we have no need to expand the local graph yet, so we set $X = \emptyset$.

Stage 2. Max-Flow Computation. Once L is correctly expanded, we compute the maximum flow f on the local graph L using any available max-flow subroutine. We then update our flow vector $F \leftarrow F + f$.

Let L_f denote the residual graph of the flow. This graph is formed by replacing the capacity c_{ij} of an edge in E_L by $c_{ij} - f_{ij}$, where f_{ij} is the flow on edge (i,j), and where the capacity of edge (j,i) is replaced with the value f_{ij}.

Stage 3. Updates. After computing a maximum flow, we resolve the effects of the flow and determine whether the local graph should be further expanded. We begin by updating the local graph to be the residual graph of f, and find the set of nodes still connected to s by a chain of unsaturated edges. We refer to this as the *source set* S. When we converge to a max flow, this is the set that is returned.

We determine the set of nodes around which to expand L in the same way LocalImprove does after computing a localized blocking flow (Orecchia & Zhu, 2014). The nodes to expand on are exactly those whose edge to t was saturated by the flow f. An edge (v,t) is saturated when the flow f_{vt} is equal to the available capacity from node v to t, so we determine the new expansion set X as follows:

$$X \leftarrow \{v \in V_L : f_{vt} = c_{vt} \text{ for } f\}.$$

If X is non-empty, there exists at least one node $x \in X$

Algorithm 1 3StageFlow

```
Input: graph $G$, parameters $\alpha, \delta$, seed set $R$
Initialize:
$V_L := \{R, \text{Neigh}(R), s, t\}$
$E_L := \{(s, R), (R, \text{Neigh}(R)), (\text{Neigh}(R), t)\}$
$F := 0; \quad X := \emptyset$
while $X \neq \emptyset$ or $F = 0$ do
  1. Expand $W$
  for $x \in X$ do
    $V_L \leftarrow V_L \cup \text{Neigh}(x)$
    $E_L \leftarrow E_L \cup \{(x, v) : v \in V_L\} \cup \{(y, t) : y \in \text{Neigh}(x)\}$
  end for
  2. Max Flow:
  $f \leftarrow \text{MaxSTflow}(L); \quad F \leftarrow F + f$
  3. Update $L$
  $L \leftarrow L_f; \quad S \leftarrow$ source set
  $X \leftarrow$ nodes whose edge to $t$ was saturated
end while
```

in the source set that has edges and neighboring nodes not yet included in L. This implies more flow could be routed from s to t through x. If X is empty, we will show in the next section that the current flow F is optimal and we no longer need to expand the local graph.

An outline for 3StageFlow is given in Algorithm 1.

4.2. Convergence of 3StageFlow

The following lemma is analogous to a result shown for the LocalImprove algorithm (Orecchia & Zhu, 2014). We use it to prove that 3StageFlow converges to a maximum s-t flow of G'.

Lemma 2 *If S is the set of nodes returned by* 3StageFlow, *then*

$$S \subseteq R \cup P_x,$$

where P_x is the set of nodes we have previously expanded on.

Proof The algorithm terminates when $X = \emptyset$ and $F > 0$. If we assume S is not a subset of $R \cup P_x$, then there exists a node $x \in S$ such that $x \notin R$ and $x \notin P_x$. Because $x \in \bar{R}$, this node must share an edge in the local graph with t. Since $x \in S$, there is a path of unsaturated edges connecting s and x, so in order for f to be maximal the edge (x, t) must be saturated. This is a contradiction, because $X = \emptyset$ implies that edge (x, t) was not saturated on the most recent iteration, and $x \notin P_x$ implies we did not previously expand on x, meaning (x, t) was not saturated in any previous iteration.■

We can now prove the optimality of the set returned by 3StageFlow.

Theorem 2 *When $X = \emptyset$ and $F > 0$, F is a maximum flow of G' and* cut$(S \cup \{s\})$ *is the minimum s-t cut set of G'.*

Proof We include the requirement $F > 0$ to indicate we have not stopped before the first iteration. In the local graph, the set of saturated edges between S and $V_L \backslash S$ defines an s-t cut with capacity equal to the total amount of flow routed from s to t. The capacity of any s-t cut is an upper bound for the amount of flow that can be routed from source to sink, so we see that F and cut$(S \cup \{s\})$ are optimal in L. By the above lemma, $S \subseteq R \cup P_x$. This implies that all neighbors and edges of S in G' are already included in the local graph L. Therefore, the max-flow and min-cut of the local graph is also an optimal flow and cut pair for the entire graph G'. ■

4.3. Strong-locality and Runtime Guarantee

The explored portion of G' directly corresponds to a subgraph of G in the following way: if we consider the local graph L and remove s and t and all edges incident to them, we end up with a subgraph of G which we call the *explored subgraph* and denote G_{exp}. This explored region is exactly the subgraph of G that our method would need to extract to create the small max-flow problems. Our algorithm not only obtains a maximum s-t flow on the entire graph, but we can show that it does so without exploring the entire graph. This result substantially sharpens a related result from Orecchia & Zhu (2014, Theorem 1a).

Theorem 3 *Given a graph G, seed set R, and locality parameter $\delta > 0$, the* 3StageFlow *procedure explores a subgraph of G satisfying the following bound:*

$$\text{vol}(G_{exp}) \leq \text{vol}(R)\left(1 + \frac{2}{\varepsilon}\right) + \partial R,$$

where $\varepsilon = f(R) + \delta$.

Proof We first bound the expanded set P_x. For any α, the maximum flow of G' is bounded above by $\alpha\,\text{vol}(R)$, the capacity of the edges leading out of s. If we expand on a node $v \in L$, it means in the previous iteration the edge $(v, t) \in E_L$ was saturated, implying that $\alpha\varepsilon d_v$ flow was routed to t. The total amount of flow through these expanded nodes therefore must satisfy

$$\textstyle\sum_{p \in P_x} \alpha\varepsilon d_p \leq \alpha\,\text{vol}(R),$$

which gives the bound

$$\text{vol}(P_x) = \textstyle\sum_{p \in P_x} d_p \leq \frac{\text{vol}(R)}{\varepsilon}.$$

By the construction and update procedure of L, the vertex set of G_{exp} is $R \cup P_x \cup Q$, where $Q = \text{Neigh}(R \cup P_x)$. This subgraph includes all edges incident to nodes in $R \cup P_x$, but contains no edges between nodes in Q, the nodes around which L has not been expanded. Because of this, the volume of Q can be upper bounded by the volume of P_x and the cut of R as follows:

$$\text{vol}(Q) = |\,\text{cut}(Q, P_x)| + |\,\text{cut}(Q, R)| \leq \text{vol}(P_x) + \partial R.$$

This can be used to show the final result:

$$\begin{aligned}\mathrm{vol}(G_{\exp}) &= \mathrm{vol}(R) + \mathrm{vol}(P_x) + \mathrm{vol}(Q)\\ &\le \mathrm{vol}(R) + 2\,\mathrm{vol}(P_x) + \partial R \le \mathrm{vol}(R)(1+\tfrac{2}{\varepsilon}) + \partial R.\end{aligned}$$

■

This result implies the following—extremely crude—strongly-local runtime guarantee. In the worst case, each flow-problem takes $O(\mathrm{vol}(R)^2/\varepsilon^2)$ to solve using the algorithm from Orlin (2013). We have at most one flow problem for each edge in the final local graph (if we grew the local graph by one vertex at a time, we must get at least one edge), giving an overall strongly-local bound of $O(\mathrm{vol}(R)^3/\varepsilon^3)$. This is highly conservative and worse than the bound on LocalImprove from Orecchia & Zhu (2014); we expect real-world runtimes to be substantially faster.

4.4. Full Outline of SimpleLocal

Given a graph G with reference set R, SimpleLocal finds a good conductance cut by repeatedly calling 3StageFlow to find the smallest α such that the maximum s-t flow of $G'_R(\alpha,\delta)$ is less than $\alpha\,\mathrm{vol}(R)$.

Algorithm 2 SimpleLocal

```
Input: $G$, $R$, locality parameter $\delta \ge 0$
$\alpha := \phi(R)$
$[F, S] :=$ 3StageFlow($G'_R(\alpha,\delta)$)
while $flow(F) < \alpha\,\mathrm{vol}(R)$ do
    $\alpha \leftarrow \phi(S)$;    $S^* \leftarrow S$
    $[F, S] :=$ 3StageFlow($G'_R(\alpha,\delta)$)
end while
Return: $S^*$
```

This procedure finds the set S^* that minimizes

$$\bar{\phi}_R(S) = \partial S/(\mathrm{vol}(R\cap S) - \varepsilon\,\mathrm{vol}(\bar{R}\cap S)), \qquad (7)$$

which is related to the relative quotient score (3).

4.5. Cut Quality Guarantee

The following result is an extension of the theorem from Andersen & Lang (2008), updated to include the effects of our parameter δ.

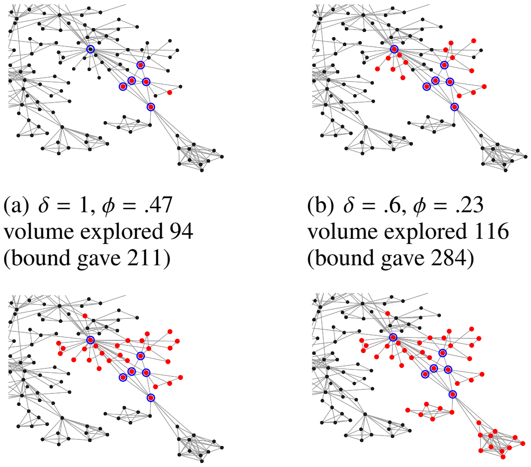
Theorem 4 *Given an initial reference set $R \subset V$ with* $\mathrm{vol}(R) \le \mathrm{vol}(\bar{R})$, SimpleLocal *returns a cut set S^* where*

1. *if $C \subseteq R$, then $\phi(S^*) \le \phi(C)$.*
2. *For all sets of nodes C such that for some $\gamma > \delta$*

$$\frac{\mathrm{vol}(R\cap C)}{\mathrm{vol}(C)} \ge \frac{\mathrm{vol}(R)}{\mathrm{vol}(V)} + \gamma\frac{\mathrm{vol}(\bar{R})}{\mathrm{vol}(V)},$$

we have $\phi(S^*) \le \frac{1}{(\gamma-\delta)}\phi(C)$.

(We include a full proof in the supplementary material.)



(a) $\delta = 1$, $\phi = .47$
volume explored 94
(bound gave 211)

(b) $\delta = .6$, $\phi = .23$
volume explored 116
(bound gave 284)

(c) $\delta = .3$, $\phi = .09$
volume explored 160
(bound gave 455)

(d) $\delta = 0$, $\phi = .03$
volume explored 522
(bound gave ∞)

Figure 3. SimpleLocal results on a small graph for different δ values where δ controls the sparsity regularization and size of the output. The reference set is given by the nodes circled in blue, and the returned set S^* is shown in red.

5. Experiments

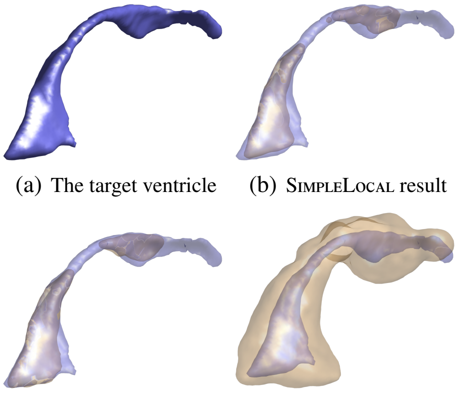
In this section we present experimental results for SimpleLocal on two graphs. We begin with an example on a small collaboration network to illustrate the effect of the locality parameter δ. We then turn our attention to graphs from MRI scans to demonstrate SimpleLocal's ability to solve problems on extremely large graphs. Our implementation of SimpleLocal and 3StageFlow are in Matlab, using Gurobi to solve the max-flow problems.

5.1. Netscience Example

Newman's netscience graph is a collaboration network with 379 nodes and a total volume of 1828. The reference set we use is a node and its immediate neighbors. We run SimpleLocal for decreasing values of δ from 1 to 0 (and implicitly, decreasing amounts of regularization) to obtain cuts near R of increasing size. These sets have increasingly better conductance. Note that for $\delta = 0$ we are computing the Improve objective. We illustrate our results in Figure 3.

5.2. MRI Scans

To demonstrate the scalability of our algorithm, we consider identifying a region in a 3d MRI scan. We obtained a labeled MRI scan from the MICCAI-2012 challenge with 256×287×256 (≈ 18 million) voxels (Marcus et al., 2007). We formed a weighted graph based on adjacent voxel similarity (see supplement for details). The final graph contained around 467 million edges and 18 million voxels.



(a) The target ventricle (b) SimpleLocal result

(c) Refined SimpleLocal (d) A spectral result

Figure 4. Results from segmenting the left lateral ventricle in an 18 million voxel MRI scan on a graph with 467 million edges. The true set is always show in blue. Our new flow algorithms track the boundary closely but cannot find the bridge in the ventricle, whereas a spectral method returns a substantially larger region.

Table 1. Statistics on the true left lateral ventricle and the sets returned by the three methods SimpleLocal, a refinement step, and spectral. The refined set gave the best conductance and highest accuracy overall.

method	size	ϕ	volume explored	precision	recall	time (sec.)
True	3965	0.129	–	–	–	–
Sim.Loc.	2425	0.089	2463247	0.96	0.59	278.4
+Refined	2737	0.067	1845966	0.97	0.67	+97.5
Spectral	27918	0.094	5280988	0.14	0.99	9.6

The left lateral ventricle is a cavity in the interior of the brain shown in Figure 4a. We use our SimpleLocal method, a one-step 3StageFlow refinement procedure (typical of what might be done in practice), and a spectral method to identify this region from 75 randomly chosen seed voxels (Figures 4b-d). (See the supplement for the details of the computations and parameter choices.) We present the statistics of the four sets in Table 1. Overall, the flow method accurately tracks the true boundary of the region, although it is unable to complete an internal bridge within the region. The refinement step fills in the region slightly more. In comparison, the spectral method returns a much larger set that contains the entire ventricle, but completely misses the boundary. This mirrors the intuition from the introduction and results on this same spectral method in community detection, where it often finds *larger, but imprecise* communities (Kloster & Gleich, 2014).

Note that the bridge of the ventricle is unlikely to be found by our method in this case. This happens because either flow-based set identified has a conductance value that is smaller (0.089 and 0.067) than the conductance of the entire region (0.129). Attempting to improve the conductance value will only shrink the identified region further (see the supplement for a few of these smaller, better conductance sets). Another curious aspect of SimpleLocal's result set is that it is disconnected. The larger of the two regions actually has a smaller conductance value itself, but the method finds a disconnected set because of the disconnected seeds. In terms of the runtime, the spectral method is faster than our sequence of max-flow problems. We discuss engineering details that could improve runtime in the supplement.

6. Conclusions and Discussion

We have given a new, simple, strongly-local algorithm for a commonly occurring problem that arises in semi-supervised learning, community detection on graphs, and image segmentation. This algorithm begins with a reference set that reflects a region of the graph known to be important and seeks a better conductance set nearby. Our method is heavily influenced by both the Improve and LocalImprove methods. In comparison with Improve, our method is strongly-local and practically scalable (given a max-flow solver for the local graphs). In comparision with LocalImprove, we have a significantly worse theoretical runtime because we solve a sequence of *maximum flow* problems compared with their use of *blocking flows*. However, our algorithm is simple to implement and can take advantage of many well-engineered maximum flow codes, such as Boykov and Kolmogorov's method that enables efficient modified flows (Boykov & Kolmogorov, 2004). We also identified the implicit source of locality in the LocalImprove method (Theorem 1), which may enable even faster methods in the future.

The new SimpleLocal implementation enabled us to run experiments on a massive MRI scan with 467 million edges that would not have been possible or desirable in a weakly-local sense using traditional graph algorithms, because the output should be a set of roughly 4000 vertices out of 18 million. Our work thus opens new possibilities in the use of maximum flows for machine learning. In particular, using a combination of spectral and flow methods will likely lead to improved results on many problems due to their complementary properties. Spectral methods can help quickly identify expanded, crude regions that the flow-based methods could contract to sharpen the boundaries.

In future work, we plan to extend our contribution to approximate maximum-flow solutions. This would enable us to take advantage of recent innovations that produce approximate maximum-flows in nearly-linear time (Christiano et al., 2011; Lee et al., 2013; Sherman, 2013)—which would likely lead to a better theoretical runtime as well. Also, we wish to better understand the tradeoffs between spectral and flow methods using this new strongly-local computational primitive.

Acknowledgments

We'd like to acknowledge and thank several funding agencies for supporting our work. Gleich was supported by NSF awards IIS-1546488, Center for Science of Information STC, CCF-093937, CAREER CCF-1149756, and DARPA SIMPLEX. Veldt was supported by NSF award IIS-1546488. Mahoney would like to acknowledge the Army Research Office, the Defense Advanced Research Projects Agency, and the Department of Energy for providing partial support for this work.

References

Andersen, Reid and Lang, Kevin. An algorithm for improving graph partitions. In *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms (SODA2008)*, pp. 651–660, January 2008.

Andersen, Reid and Lang, Kevin J. Communities from seed sets. In *Proceedings of the 15th international conference on the World Wide Web*, pp. 223–232, 2006. doi: 10.1145/1135777.1135814.

Andersen, Reid, Chung, Fan, and Lang, Kevin. Local graph partitioning using PageRank vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

Blum, Avrim and Chawla, Shuchi. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 19–26, 2001.

Boykov, Yuri and Kolmogorov, Vladimir. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September 2004. doi: 10.1109/TPAMI.2004.60.

Brindle, Nick and Zhu, Xiaojin. p-voltages: Laplacian regularization for semi-supervised learning on high-dimensional data. Workshop on Mining and Learning with Graphs (MLG2013), 2013.

Christiano, Paul, Kelner, Jonathan A., Madry, Aleksander, Spielman, Daniel A., and Teng, Shang-Hua. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pp. 273–282, 2011. doi: 10.1145/1993636.1993674.

Chung, Fan. Random walks and local cuts in graphs. *Linear Algebra and its Applications*, 423(1):22 – 32, 2007. doi: 10.1016/j.laa.2006.07.018.

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

Flake, Gary William, Lawrence, Steve, and Giles, C. Lee. Efficient identification of web communities. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pp. 150–160, 2000. doi: 10.1145/347090.347121.

Fujiwara, Yasuhiro and Irie, Go. Efficient label propagation. In Jebara, Tony and Xing, Eric P. (eds.), *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 784–792. JMLR Workshop and Conference Proceedings, 2014.

Gallo, G., Grigoriadis, M., and Tarjan, R. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989. doi: 10.1137/0218003.

Gleich, David F. and Mahoney, Michael M. Algorithmic anti-differentiation: A case study with min-cuts, spectral, and flow. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1018–1025, 2014.

Gleich, David F. and Mahoney, Michael W. Using local spectral methods to robustify graph-based learning algorithms. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 359–368, 2015. doi: 10.1145/2783258.2783376.

He, Zhengyu and Hong, Bo. Dynamically tuned push-relabel algorithm for the maximum flow problem on cpu-gpu-hybrid platforms. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pp. 1–10, April 2010. doi: 10.1109/IPDPS.2010.5470401.

Jeub, Lucas G. S., Balachandran, Prakash, Porter, Mason A., Mucha, Peter J., and Mahoney, Michael W. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Phys. Rev. E*, 91:012821, January 2015. doi: 10.1103/PhysRevE.91.012821.

Joachims, Thorsten. Transductive learning via spectral graph partitioning. In *ICML*, pp. 290–297, 2003.

Kloster, Kyle and Gleich, David F. Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1386–1395, 2014. doi: 10.1145/2623330.2623706.

Kloumann, Isabel M. and Kleinberg, Jon M. Community membership identification from small seed sets. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1366–1375, 2014. doi: 10.1145/2623330.2623621.

Lang, Kevin. Fixing two weaknesses of the spectral method. In Weiss, Y., Schölkopf, B., and Platt, J. (eds.), *Advances in Neural Information Processing Systems 18 (NIPS2005)*, pp. 715–722, 2005.

Lang, Kevin and Rao, Satish. A flow-based method for improving the expansion or conductance of graph cuts. In *Integer Programming and Combinatorial Optimization*, volume 3064 of *Lecture Notes in Computer Science*, pp. 325–337. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-540-25960-2_25.

Lee, Yin Tat, Rao, Satish, and Srivastava, Nikhil. A new approach to computing maximum flows using electrical flows. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pp. 755–764, 2013. doi: 10.1145/2488608.2488704.

Leskovec, Jure, Lang, Kevin J., Dasgupta, Anirban, and Mahoney, Michael W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, September 2009. doi: 10.1080/15427951.2009.10129177.

Lu, Zhiwu and Peng, Yuxin. Image annotation by semantic sparse recoding of visual content. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pp. 499–508, 2012. doi: 10.1145/2393347.2393418.

Mahoney, Michael W., Orecchia, Lorenzo, and Vishnoi, Nisheeth 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.

Marcus, Daniel S., Wang, Tracy H., Parker, Jamie, Csernansky, John G., Morris, John C., and Buckner, Randy L. 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. doi: 10.1162/jocn.2007.19.9.1498. The MRI scans originated with the OASIS project and labeled data was provided by Neuromorphometrics, Inc. `neuromorphometrics.com` under an academic subscription.

Narasimhan, M. and Bilmes, J. Local search for balanced submodular clusterings. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pp. 981–986, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

Orecchia, Lorenzo and Zhu, Zeyuan Allen. Flow-based algorithms for local graph clustering. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms*, SODA2014, pp. 1267–1286, 2014.

Orlin, James B. Max flows in o(nm) time, or better. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pp. 765–774, 2013. doi: 10.1145/2488608.2488705.

Patkar, Sachin B. and Narayanan, H. Improving graph partitions using submodular functions. *Discrete Applied Mathematics*, 131(2):535 – 553, 2003. doi: 10.1016/S0166-218X(02)00472-9.

Sherman, Jonah. Nearly maximum flows in nearly linear time. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pp. 263–269, Washington, DC, USA, 2013. IEEE Computer Society. doi: 10.1109/FOCS.2013.36.

Spielman, Daniel A. and Teng, Shang-Hua. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013. doi: 10.1137/080744888.

Zhou, Dengyong, Bousquet, Olivier, Lal, Thomas Navin, Weston, Jason, and Schölkopf, Bernhard. Learning with local and global consistency. In *NIPS*, 2003.

Zhou, Xueyuan and Srebro, Nathan. Error analysis of laplacian eigenmaps for semi-supervised learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 901–908, 2011. JMLR W&CP.

Zhu, Xiaojin, Ghahramani, Zoubin, and Lafferty, John. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pp. 912–919, 2003.

Zhu, Zeyuan Allen, Lattanzi, Silvio, and Mirrokni, Vahab. A local algorithm for finding well-connected clusters. In *Proceedings of the 30th International Conference on Machine Learning*, ICML2013, pp. 396–404, 2013.