Anti-differentiating approximation algorithms: A case study with min-cuts, spectral, and flow

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

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

Abstract

We formalize and illustrate the general concept of *algorithmic anti-differentiation*: given an algorithmic procedure, *e.g.*, an approximation algorithm for which worst-case approximation guarantees are available or a heuristic that has been engineered to be practically-useful but for which a precise theoretical understanding is lacking, an algorithmic anti-derivative is a precise statement of an optimization problem that is exactly solved by that procedure. We explore this concept with a case study of approximation algorithms for finding locally-biased partitions in data graphs, demonstrating connections between min-cut objectives, a personalized version of the popular PageRank vector, and the highly effective "push" procedure for computing an approximation to personalized PageRank. We show, for example, that this latter algorithm solves (exactly, but implicitly) an ℓ_1-regularized ℓ_2-regression problem, a fact that helps to explain its excellent performance in practice. We expect that, when available, these implicit optimization problems will be critical for rationalizing and predicting the performance of many approximation algorithms on realistic data.

1. Introduction

In an ideal setting, one begins with a well-defined objective function and one develops (or calls as a subroutine) an algorithm to solve that problem "exactly." In many applications, however, the appropriate objective function might not be known in advance and/or one might have only an approximation algorithm available for that objective. In addition, in practical settings, one might heuristically modify the algorithm—*e.g*, stop at fewer than the theoretically-appropriate number of iterations, truncate very small entries of a large dense vector to zero, etc.—in order to achieve improved performance. In these cases, it can be difficult to have a precise theoretical understanding of what these heuristic modifications are doing, even though anecdotal evidence suggests that these modifications can be the prime determinants of the practical performance of many machine learning algorithms on realistic data.

To address these issues, we would like to introduce the term *algorithmic anti-differentiation* for the following activity: analyzing a given algorithmic procedure (typically, an approximation algorithm or a heuristic) and reinterpreting it as a scheme that exactly solves a different (but, of course, related) optimization problem (implicitly, in the sense that the actual problem is never written down and may not even be known *a priori*). We have chosen the term "anti-differentiation" because many algorithmic procedures arise as a means to solve exactly the optimality conditions of some optimization problem. If, instead, we begin with a practically-useful algorithmic procedure, the idea behind algorithmic anti-differentiation is to find an optimization problem for which that procedure exactly solves the the optimality conditions. Here, we are drawing an analogy between the optimality conditions as the derivative of the optimization problem; hence, reversing the procedure is "anti-differentiation." In a simple unconstrained optimization problem, this analogy is precise, as the optimality conditions are statements about the derivative of the objective function.[1]

In this paper, we present a "case study" of algorithmic anti-differentiation, with a focus on two related problems having

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

[1]The analogy also suggests that, just as computing closed-form anti-derivatives of general functions is much harder than computing derivatives, one should expect that finding a "nice" expression for the algorithmic anti-derivative of an arbitrary approximation heuristic should be much harder than computing an optimum of a function.

to do with finding locally-biased graph partitions. We start by showing (in Section 3.1) that the solution of a PageRank problem (Page et al., 1999) is a 2-norm variation of a 1-norm formulation of a min-cut linear program related to the so-called FlowImprove procedure (Andersen & Lang, 2008). Although a similar fact was known for personalized PageRank and a locally-biased spectral partitioning problem (Mahoney et al., 2012), our new result permits us to reverse the relationships and demonstrate (in Section 3.2) a min-cut/max-flow problem that relaxes to the standard PageRank problem with uniform teleportation.

We then show (in Section 3.3) that a particularly efficient procedure for solving a personalized PageRank problem (Andersen et al., 2006) implicitly corresponds to adding a 1-norm regularization term to the 2-norm PageRank objective. By examining the optimality conditions for this regularized problem, we will be able to understand why this procedure gives rise to both *sparse solutions* and a *sparse truncated residual*; and our analysis also makes a hidden tolerance parameter in the algorithmic procedure an explicit feature of our approach.

The utility of these results is that we can begin to understand more precisely the implicit side-effects of heuristic design decisions that are often made in implementing algorithms. We illustrate this in Section 4. For example, we compare these procedures to illustrate subtle but important differences between using max-flow/min-cut programs, their spectral relatives, and the "truncated" versions of their spectral relatives; and we point out as a remark that we can also use this same setup to understand other related diffusion-based methods that have been popular recently in machine learning.

Far from inventing the concept of algorithmic anti-differentiation, our contribution is to make it precise and to provide a detailed case study of it for a class of algorithms that has received a great deal of interest recently in machine learning and data analysis. For instance, exploiting algorithmic anti-differentiation ideas has been very fruitful for understanding the relationships between various iterative methods for solving linear systems of equations in linear algebra and scientific computing. As an example of this, Saunders (1995) (in his Results 8 and 9) demonstrates an equivalence relationship between the iterates of two widely-used algorithms. Although these problems are shown to be mathematically equivalent in exact arithmetic, the numerical properties—and thus the performance in practical implementations—of these two methods differ substantially in the presence of "noise" introduced by roundoff error; and thus one variant is much preferable in practice. (For this and related reasons alluded to below, we expect that precisely understanding algorithmic anti-differentiation will help in the development of better variants of many popular machine learning algorithms in very large-scale settings.)

We conclude this introduction with a brief survey of several prior examples of studying approximation algorithms to elicit surprising optimization properties. While not exhaustive, these are the examples that most informed our approach; and they are most strongly related to instances of spectral methods, cuts, and flows. First, Dhillon et al. (2007) show that the kernel k-means algorithm is equivalent to a particular type of trace minimization, and they used this insight to produce a scalable graph clustering method based on the normalized cut objective. Subsequently, Kulis & Jordan (2012) construct a Bayesian algorithmic anti-derivative, which gives rise to the Dirichlet process-means (DP-means) algorithm. Second, Koutra et al. (2011) show how an algorithmic approximation to a set of Belief Propagation equations corresponds to solving a linear system that is closely related to a diffusion process. Third, Chin et al. (2013) use the relationship between LASSO-style objectives and max-flow/min-cut problems (of the form of Prob. (2) below) and shortest paths problems in order to derive new runtime bounds for the LASSO problems that occur in machine learning (Tibshirani, 1994). Finally, the work that is perhaps most-closely related to ours shows how algorithmic decisions such as early stopping (of an iterative algorithm) can be recast as linear operators that solve particular regularized SDPs (Orecchia & Mahoney, 2011). This has the interpretation that approximate computation *in and of itself* can implicitly implement regularization (*e.g.*, in the "space" of approximation algorithms (Mahoney, 2012)), whereas our results below establish similar results for the *solutions* of those approximation algorithms.

2. Notation and background

To draw precise relationships between various formulations of min-cut/max-flow problems on graphs, spectral variations of these problems, and related PageRank problems, we will require an unusually high degree of precision in our notation. Let $G = (V, E, w)$ be a *connected, undirected graph with positive edge weights*, and let $n = |V|$ be the number of vertices. Fix an indexing of the vertices from 1 to n, and then the adjacency matrix is the $n \times n$ matrix $\mathbf{A}$ where

$$A_{i,j} = \begin{cases} w(i,j) & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The degrees of each node in G are given by the row-sums of the matrix $\mathbf{A}$. Others have called these the weighted degrees, but since all of our graphs are weighted, we won't make this distinction. The matrix $\mathbf{D} = \text{diag}(\mathbf{Ae})$, *i.e.*, $\mathbf{D}$ is a $n \times n$ square diagonal matrix with the degree of each vertex on the diagonal; the vector $\mathbf{d} = \mathbf{De}$ is the vector of degrees.

The vector $\mathbf{e}$ will denote the all-ones vector; and $\mathbf{e}_i$ will denote the vector with a one in the ith position and zeros everywhere else. In addition, for a set S, $\mathbf{e}_S = \sum_{i \in S} \mathbf{e}_i$; $\mathbf{I}_S$ are columns of the identity matrix for vertices in S in a fixed order; and $\text{vol}(S)$ is $\mathbf{e}_S^T \mathbf{d}$, the sum of degrees of nodes in S.

Let m be the number of undirected edges in G (*i.e.*, where each edge is only counted once), and fix an ordering of these m edges. The edge-node incidence matrix of G is an $m \times n$ matrix $\mathbf{B}$ where each row is of the form $(\mathbf{e}_i - \mathbf{e}_j)^T$ for an edge (i, j). Note that the incidence matrix does *not* include any effect of the edge weights and includes only the combinatorial structure of the graph. We let the diagonal matrix $\mathbf{C}$ hold the information on the edge weights, where the order of edges is the same. In particular, we use the notation $C_{(i,j)}$ to denote a diagonal of this matrix for the edge (i, j). In this case, the combinatorial Laplacian matrix is given by $\mathbf{L} = \mathbf{B}^T\mathbf{C}\mathbf{B} = \mathbf{D} - \mathbf{A}$; and the uniform random-walk transition matrix on G is given by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$.

As an aside, it is tempting to define the incidence matrix as "$\mathbf{C}^{-1/2}\mathbf{B}$" so as to preserve the relationship: "$\mathbf{L} = \mathbf{B}^T\mathbf{B}$". This choice would end up causing confusion when we begin to discuss how PageRank and related diffusion computations are related to the 1-norm min-cut problems. We have found it most convenient to treat the edge weights via a weighted norm, such that if $\mathbf{x}$ is a binary vector, then

$$\begin{aligned} \|\mathbf{B}\mathbf{x}\|_{C,1} &= \sum_{(i,j) \in E} C_{(i,j)} |x_i - x_j| \\ &= \text{cut}(S), \quad \text{where} \quad S = \{i : x_i = 1\}. \end{aligned}$$

The spectral problem will then replace the 1-norm with a 2-norm, but it will preserve the weighting so that:

$$\|\mathbf{B}\mathbf{x}\|_{C,2}^2 = \sum_{(i,j) \in E} C_{(i,j)} (x_i - x_j)^2.$$

The PageRank problem for an undirected graph can be defined as the solution of the linear equation (Arasu et al., 2002; Langville & Meyer, 2006):

$$(\mathbf{I} - \beta \mathbf{P}^T)\mathbf{x} = (1 - \beta)\mathbf{v}, \tag{1}$$

where $\mathbf{P}$ is the random walk matrix for the graph (as defined above), β is the teleportation parameter which satisfies $0 < \beta < 1$, and $\mathbf{v}$ is the non-negative teleportation distribution vector with $v_i \geq 0$ and $\sum_i v_i = \mathbf{e}^T\mathbf{v} = 1$. This formulation is entirely equivalent to the standard definition as the stationary distribution of a random walk with restart or the PageRank Markov chain (Langville & Meyer, 2006).

Recall also that, for an undirected graph, the following formulations of PageRank are all equivalent:

1. $(\mathbf{I} - \beta \mathbf{A}\mathbf{D}^{-1})\mathbf{x} = (1 - \beta)\mathbf{v}$;
2. $(\mathbf{I} - \beta \mathcal{A})\mathbf{y} = (1 - \beta)\mathbf{D}^{-1/2}\mathbf{v}$,
   where $\mathcal{A} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\mathbf{x} = \mathbf{D}^{1/2}\mathbf{y}$; and
3. $[\alpha \mathbf{D} + \mathbf{L}]\mathbf{z} = \alpha \mathbf{v}$ where $\beta = 1/(1 + \alpha)$ and $\mathbf{x} = \mathbf{D}\mathbf{z}$.

All of these relationships are straightforward to derive from the original PageRank equation. We will primarily use the first and third; but we note that the second arises in a setup of semi-supervised learning on a graph (Zhou et al., 2004), a topic upon which we will comment below.

3. Main theoretical results

In this section, we present two algorithmic anti-derivatives. The first (in Section 3.1) relates the PageRank problem on an undirected graph to a particular minimum s, t-cut construction. Under certain conditions, this problem is a 2-norm minorant of a 1-norm formulation of the min-cut objective. We also (in Section 3.2) establish the reverse relationship that extracts a cut/flow problem from *any* PageRank problem. The second (in Section 3.3) relates an approximation algorithm for the personalized PageRank problem to a 1-norm regularized version of the 2-norm objective.

3.1. Relating min-cut to PageRank

Recall the 1-norm formulation of a linear program for the min-s, t-cut problem:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}\mathbf{x}\|_{C,1} \\ \text{subject to} & x_s = 1, x_t = 0, \mathbf{x} \geq 0. \end{array} \tag{2}$$

When the weights are integers, these problems are usually solved using an efficient max-flow routine in order to compute a strictly-integral solution. Here, we consider a specific s, t-cut problem inspired by the FlowImprove procedure (Andersen & Lang, 2008), which was also used in recent work on a local version of this objective (Orecchia & Zhu, 2014). In order to state this problem, we must fix a set of vertices S. These roughly correspond to the teleportation vector in the PageRank problem, and we make this analogy precise below. Once we have this set, we define a new weighted graph based on the original graph that we call the *localized cut graph*. This graph consists of the original graph, with two additional vertices, s and t, that connect to S and $\bar{S}$, respectively, with weights equal to α multiplied by the degree of each connected vertex. We illustrate a localized cut graph in Figure 1. We now state the formal version for completeness.

Definition 1 Let $G = (V, E)$ be a graph, let S be a set of vertices, possibly empty, let $\bar{S}$ be the complement set, and let α be a non-negative constant. Then the *localized cut graph* is the weighted, undirected graph with adjacency matrix:

$$\mathbf{A}_S = \begin{bmatrix} 0 & \alpha \mathbf{d}_S^T & 0 \\ \alpha \mathbf{d}_S & \mathbf{A} & \alpha \mathbf{d}_{\bar{S}} \\ 0 & \alpha \mathbf{d}_{\bar{S}}^T & 0 \end{bmatrix},$$

where $\mathbf{d}_S = \mathbf{D}\mathbf{e}_S$ is a degree vector localized on the set S, $\mathbf{A}$ is the adjacency matrix of the original graph G, and $\alpha \geq 0$

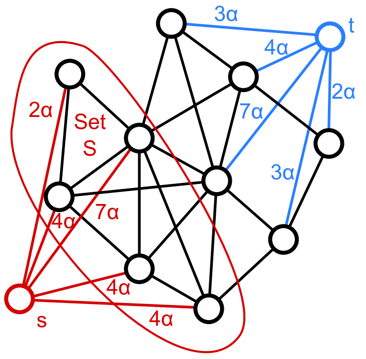

Figure 1. An illustration of a *localized cut graph* used in our framework. The vertices in the set S are circled. We will use this graph to illustrate our optimization problems.

is a non-negative weight. Note that the first vertex is s and the last vertex is t.

In the remainder of the section, we'll use the α and S parameter to denote the matrices for the localized cut graph. For example, $\mathbf{B}(S)$ is the incidence matrix of the localized cut graph, which depends on the set S:

$$\mathbf{B}(S) = \begin{bmatrix} \mathbf{e} & -\mathbf{I}_S & 0 \\ 0 & \mathbf{B} & 0 \\ 0 & -\mathbf{I}_{\bar{S}} & \mathbf{e} \end{bmatrix},$$

where, recall, the variable $\mathbf{I}_S$ are the columns of the identity matrix corresponding to vertices in S. The edge-weights of the localized cut graph are given by the diagonal matrix $\mathbf{C}(\alpha)$, which depends on the value α. In this case, the minimum weighted s, t cut in the flow graph solves the linear program:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}(S)\mathbf{x}\|_{C(\alpha),1} \\ \text{subject to} & x_s = 1, x_t = 0, \mathbf{x} \geq 0. \end{array} \tag{3}$$

The following theorem is our first algorithmic anti-derivative. In it, we show that PageRank implicitly solves a 2-norm variation of the 1-norm formulation of the s, t-cut problem, as given in Prob. (3). (Recall that the 2-norm is a minorant of the 1-norm.)

Theorem 1 *Let $\mathbf{B}(S)$ be the incidence matrix for the localized cut graph, and $\mathbf{C}(\alpha)$ be the edge-weight matrix. The PageRank vector $\mathbf{z}$ that solves*

$$(\alpha\mathbf{D} + \mathbf{L})\mathbf{z} = \alpha\mathbf{v}$$

with $\mathbf{v} = \mathbf{d}_S/vol(S)$ is a renormalized solution of the 2-norm cut computation:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}(S)\mathbf{x}\|_{C(\alpha),2} \\ \textit{subject to} & x_s = 1, x_t = 0. \end{array} \tag{4}$$

Specifically, if $\mathbf{x}(\alpha, S)$ is the solution of Prob. (4), then

$$\mathbf{x}(\alpha, S) = \begin{bmatrix} 1 \\ vol(S)\mathbf{z} \\ 0 \end{bmatrix}.$$

Proof This result is mainly algebraic. The key idea is that the 2-norm problem corresponds with a quadratic objective, which PageRank solves. The quadratic objective for the 2-norm approximate cut is:

$$\begin{aligned} &\|\mathbf{B}(S)\mathbf{x}\|^2_{C(\alpha),2} \\ &\quad = \mathbf{x}^T\mathbf{B}(S)^T\mathbf{C}(\alpha)\mathbf{B}(S)\mathbf{x} \\ &\quad = \mathbf{x}^T \begin{bmatrix} \alpha\text{vol}(S) & -\alpha\mathbf{d}_S^T & 0 \\ -\alpha\mathbf{d}_S & \mathbf{L} + \alpha\mathbf{D} & -\alpha\mathbf{d}_{\bar{S}} \\ 0 & -\alpha\mathbf{d}_{\bar{S}} & \alpha\text{vol}(\bar{S}) \end{bmatrix} \mathbf{x}. \end{aligned}$$

If we apply the constraints that $x_s = 1$ and $x_t = 0$ and let $\mathbf{x}_G$ be the free set of variables, then we arrive at the unconstrained objective:

$$\begin{aligned} &\begin{bmatrix} 1 & \mathbf{x}_G^T & 0 \end{bmatrix} \begin{bmatrix} \alpha\text{vol}(S) & -\alpha\mathbf{d}_S^T & 0 \\ -\alpha\mathbf{d}_S & \mathbf{L} + \alpha\mathbf{D} & -\alpha\mathbf{d}_{\bar{S}} \\ 0 & -\alpha\mathbf{d}_{\bar{S}} & \alpha\text{vol}(\bar{S}) \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{x}_G \\ 0 \end{bmatrix} \\ &\quad = \mathbf{x}_G^T(\mathbf{L} + \alpha\mathbf{D})\mathbf{x}_G - 2\alpha\mathbf{x}_G^T\mathbf{d}_S + \alpha\text{vol}(S). \end{aligned}$$

Here, the solution $\mathbf{x}_G$ solves the linear system

$$(\alpha\mathbf{D} + \mathbf{L})\mathbf{x}_G = \alpha\mathbf{d}_S.$$

The vector $\mathbf{x}_G = \text{vol}(S)\mathbf{z}$, where $\mathbf{z}$ is the solution of the PageRank problem defined in the theorem, which concludes the proof.

3.2. Relating PageRank to min-cut

The result of the previous subsection gives only one "direction" of the relationship between the PageRank problem and the min-cut problem. That is, there is a cut/flow problem that gives rise to a PageRank problem. In this subsection, we establish the reverse relationship that extracts a cut/flow problem from *any* PageRank problem. This result is of interest by itself, but it is also of interest as a precursor to our main result in the next subsection.

Notice that the reason the proof of Theorem 1 works is that the edges we added had weights proportional to the degree of the node, and hence the increase to the degree of the nodes was proportional to their current degree. This property, in turn, causes the diagonal of the Laplacian matrix of the localized cut graph to become $\alpha\mathbf{D} + \mathbf{D}$. This idea forms the basis of our subsequent analysis. For a general PageRank problem, however, we require a slightly more general definition of the localized cut graph, which we call a *PageRank cut graph.*

Definition 2 Let $G = (V, E)$ be a graph, and let $\mathbf{s} \geq 0$ be a vector such that $\mathbf{d} - \mathbf{s} \geq 0$. Let s connect to each node in G with weights given by the vector $\alpha\mathbf{s}$, and let t connect to each node in G with weights given by $\alpha(\mathbf{d} - \mathbf{s})$. Then the *PageRank cut graph* is the weighted, undirected graph with adjacency matrix:

$$\mathbf{A}(\mathbf{s}) = \begin{bmatrix} 0 & \alpha\mathbf{s}^T & 0 \\ \alpha\mathbf{s} & \mathbf{A} & \alpha(\mathbf{d}-\mathbf{s}) \\ 0 & \alpha(\mathbf{d}-\mathbf{s})^T & 0 \end{bmatrix}.$$

We use $\mathbf{B}(\mathbf{s})$ to refer to the incidence matrix of this PageRank cut graph; and note that when $\mathbf{s} = \mathbf{d}_S$, then we return to the localized cut graph definition.

With this, we state the following theorem, which is a sort of converse to Theorem 1, as well as a corollary of independent interest. Due to its similarity with the proof of Theorem 1, the proof of this theorem is omitted.

Theorem 2 *Consider any PageRank problem that fits the framework of Prob.* (1). *The PageRank vector* $\mathbf{z}$ *that solves*

$$(\alpha\mathbf{D} + \mathbf{L})\mathbf{z} = \alpha\mathbf{v}$$

is a renormalized solution of the 2-norm cut computation:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}(\mathbf{s})\mathbf{x}\|_{C(\alpha),2} \\ \textit{subject to} & x_s = 1, x_t = 0 \end{array} \tag{5}$$

$\mathbf{s} = \mathbf{v}$. *Specifically, if* $\mathbf{x}(\alpha, S)$ *is the solution of the 2-norm cut, then*

$$\mathbf{x}(\alpha, \mathbf{s}) = \begin{bmatrix} 1 \\ \mathbf{z} \\ 0 \end{bmatrix}.$$

Corollary 1 *If* $\mathbf{s} = \mathbf{e}$, *then the solution of a 2-norm cut is a reweighted, renormalized solution of PageRank with* $\mathbf{v} = \mathbf{e}/n$.

That is, as a corollary of our framework, the *standard* PageRank problem with $\mathbf{v} = \mathbf{e}/n$ gives rise to a cut problem where s connects to each node with weight α and t connects to each node v with weight $\alpha(d_v - 1)$.

Remark 1 With respect to solving an objective, *e.g.*, of the form of Prob. (3), note that the weights $C(\alpha)$ will *not* in general be integer-valued. In theory, any max-flow/min-cut solver will compute the correct solution for these cases; but we have observed poor behavior from most implementations of the efficient push-relabel method (Goldberg & Tarjan, 1988). The most reliable way to solve these problems for general non-integer weights is to use a commercial linear programming package. Another alternative is to scale and round the weights back to integers. This latter approach introduces an arbitrary small error and may demand many bits of precision in the integers.

Remark 2 This machinery we have introduced will produce other diffusion equations as well, depending on the details of the setup, which may be of independent interest. For example, consider the equally natural setting where the edges from s to the graph and the edges from the graph to t are not degree-weighted, but we've reweighted the graph instead:

$$\begin{bmatrix} 0 & \mathbf{e}_S^T & 0 \\ \mathbf{e}_S & \theta\mathbf{A} & \mathbf{e}_{\bar{S}} \\ 0 & \mathbf{e}_{\bar{S}} & 0 \end{bmatrix}.$$

In this case, the 2-norm approximate cut solution on vertices of the graph solves:

$$(\mathbf{I} + \theta\mathbf{L})\mathbf{x} = \mathbf{e}_S.$$

Remark 3 These ideas are likely applicable much more generally in diffusion-based machine learning. Recall, *e.g.*, that the procedure of Zhou et al. (2004) for semi-supervised learning on graphs solves the following:

$$(\mathbf{I} - \beta\mathcal{A})^{-1}\mathbf{Y}.$$

This is exactly a PageRank equation for a degree-based scaling of the labels, and thus our construction from Theorem 2 is directly applicable.

3.3. Relating approximate PageRank and exact 1-norm regularization

Next, we show that the Andersen, Chung, Lang (ACL) procedure for *approximating* a personalized PageRank vector (Andersen et al., 2006) (of the form considered in Section 3.1) *exactly* computes a hybrid 1-norm 2-norm variant of the min-cut problem. The balance between these two terms has the effect of producing sparse PageRank solutions that also have sparse truncated residuals, and it also provides an interesting connection with ℓ_1-regularized ℓ_2-regression problems. We start by reviewing the ACL method.

Consider the personalized PageRank problem $(\mathbf{I} - \beta\mathbf{A}\mathbf{D}^{-1})\mathbf{x} = (1 - \beta)\mathbf{v}$, where $\mathbf{v} = \mathbf{e}_i$ is localized onto a single node. If $\mathbf{A}$ is a connected, undirected graph, then $\mathbf{x}$ a strictly positive solution vector. ACL use an algorithmic procedure to approximate this personalized PageRank vector with a bounded amount of work. In addition to the PageRank parameter β, the procedure has two parameters: $\tau > 0$ is a accuracy parameter that determines when to stop, and $0 < \rho \leq 1$ is an additional approximation term that we introduce. As $\tau \to 0$, the computed solution $\mathbf{x}$ goes to the personalized PageRank vector that is non-zero everywhere. The value of ρ has been $1/2$ in most previous implementations of the procedure. Here, we present a modified procedure that differs slightly from their original algorithm that makes the effect of ρ explicit.[2]

[2]The procedure we describe is based on an implementation provided by Reid Andersen and matches the procedure used in the large-scale empirical study (Leskovec et al., 2009).

```
1. $\mathbf{x}^{(1)} = 0, \mathbf{r}^{(1)} = (1-\beta)\mathbf{e}_i, k = 1$
2. while any $r_j > \tau d_j$        $d_j$ is the degree of node $j$
3.     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + (r_j - \tau d_j \rho)\mathbf{e}_j$
4.     $\mathbf{r}_i^{(k+1)} = \begin{cases} \tau d_j \rho & i = j \\ r_i^{(k)} + \beta(r_j - \tau d_j \rho)/d_j & i \sim j \\ r_i^{(k)} & \text{otherwise} \end{cases}$
5.     $k \leftarrow k + 1$
```

One of the important properties of this procedure is that the algorithm maintains the invariant $\mathbf{r} = (1-\beta)\mathbf{v} - (\mathbf{I} - \beta\mathbf{A}\mathbf{D}^{-1})\mathbf{x}$ throughout.[3] This method is closely related to the Gauss-Seidel method, which is a coordinate descent method for solving the personalized PageRank linear system, except for a few key differences. First, the algorithm only takes a partial step along the gradient in each coordinate. Second, the algorithm does not cycle through all the rows like the standard Gauss-Seidel procedure; instead, it just picks any row with a large residual (the original method used a queue of valid vertices).

For any $0 \le \rho \le 1$, this algorithm converges because the sum of entries in the residual always decreases monotonically. At the solution we will have

$$0 \le \mathbf{r} \le \tau \mathbf{d},$$

which provides an ∞-norm style worst-case approximation guarantee to the exact PageRank solution.

Our main result in this section establishes a precise algorithmic anti-derivative for the ACL procedure, in the case that $\rho = 1$. In the same way that Theorem 2 establishes that a PageRank vector can be interpreted as optimizing an ℓ_2 objective involving the edge-incidence matrix, the following theorem establishes that the ACL procedure to approximate this vector can be interpreted as solving an ℓ_1-regularized ℓ_2 objective.

Theorem 3 *Fix a subset of vertices S. Let $\mathbf{x}$ be the output from the Andersen, Chung, Lang procedure with $\rho = 1$, $0 < \beta < 1$, $\mathbf{v} = \mathbf{d}_S/vol(S)$, and τ fixed. Set $\alpha = \frac{1-\beta}{\beta}$, $\kappa = \tau vol(S)/\beta$, and let $\mathbf{z}_G$ be the solution on graph vertices of the sparsity-regularized cut problem:*

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{B}(S)\mathbf{z}\|^2_{C(\alpha),2} + \kappa\|\mathbf{D}\mathbf{z}\|_1 \\ \textit{subject to} & z_s = 1, z_t = 0, \mathbf{z} \ge 0 \end{array}, \tag{6}$$

where $\mathbf{z} = \begin{bmatrix} 1 \\ \mathbf{z}_G \\ 0 \end{bmatrix}$ as above. Then $\mathbf{x} = \mathbf{D}\mathbf{z}_G/vol(S)$.

Proof If we expand the objective function and apply the constraint $z_s = 1, z_t = 0$, Prob. (6) becomes:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{z}_G^T(\alpha\mathbf{D} + \mathbf{L})\mathbf{z}_G - \alpha\mathbf{z}_G^T\mathbf{d}_S \\ & \quad + \alpha^2 \text{vol}(S) + \kappa\mathbf{d}^T\mathbf{z}_G \\ \text{subject to} & \mathbf{z}_G \ge 0 \end{array}.$$

[3] In our description, we assumed the graph had no self-loops and fixed $\mathbf{v} = \mathbf{e}_i$. It is easy to adapt to the case of self-loops or general $\mathbf{v}$ by maintaining this invariant.

Consider the optimality conditions of this quadratic problem (where $\mathbf{s}$ are the Lagrange multipliers):

$$\begin{aligned} 0 &= (\alpha\mathbf{D} + \mathbf{L})\mathbf{z}_G - \alpha\mathbf{d}_{\bar{S}} + \kappa\mathbf{d} - \mathbf{s} \\ \mathbf{s} &\ge 0 \\ \mathbf{z}_G &\ge 0 \\ \mathbf{z}_G^T\mathbf{s} &= 0. \end{aligned}$$

These are both necessary and sufficient because $(\alpha\mathbf{D} + \mathbf{L})$ is positive definite. In addition, and for the same reason, the solution is unique.

In the remainder of the proof, we demonstrate that vector $\mathbf{x}$ produced by the ACL method satisfies these conditions. To do so, we first translate the optimality conditions to the equivalent PageRank normalization:

$$\begin{aligned} 0 = (\mathbf{I} - \beta\mathbf{A}\mathbf{D}^{-1})\mathbf{D}\mathbf{z}_G/\text{vol}(S) - \\ (1-\beta)\mathbf{d}_S/\text{vol}(S) + \beta\kappa/\text{vol}(S)\mathbf{d} - \beta\mathbf{s}/\text{vol}(S) \\ \mathbf{s} \ge 0 \qquad \mathbf{z}_G \ge 0 \qquad \mathbf{z}_G^T\mathbf{s} = 0. \end{aligned}$$

When the ACL procedure finishes with β, ρ, and τ as in the theorem, the vectors $\mathbf{x}$ and $\mathbf{r}$ satisfy:

$$\begin{aligned} &\mathbf{r} = (1-\beta)\mathbf{v} - (\mathbf{I} - \beta\mathbf{A}\mathbf{D}^{-1})\mathbf{x} \\ &\mathbf{x} \ge 0 \\ &0 \le \mathbf{r} \le \tau\mathbf{d} = \beta\kappa/\text{vol}(S)\mathbf{d}. \end{aligned}$$

Thus, if we set $\mathbf{s}$ such that $\beta\mathbf{s}/\text{vol}(S) = \beta\kappa/\text{vol}(S)\mathbf{d} - \mathbf{r}$, then we satisfy the first condition with $\mathbf{x} = \mathbf{D}\mathbf{z}_G/\text{vol}(S)$. All of these transformations preserve $\mathbf{x} \ge 0$ and $\mathbf{z}_G \ge 0$. Also, because $\tau\mathbf{d} \ge \mathbf{r}$, we also have $\mathbf{s} \ge 0$. What remains to be shown is $\mathbf{z}_G^T\mathbf{s} = 0$.

Here, we show $\mathbf{x}^T(\tau\mathbf{d} - \mathbf{r}) = 0$, which is equivalent to the condition $\mathbf{z}_G^T\mathbf{s} = 0$ because the non-zero structure of the vectors is identical. Orthogonal non-zero structure suffices because $\mathbf{z}_G\mathbf{s} = 0$ is equivalent to either $x_i = 0$ or $\tau d_i - r_i = 0$ (or both) for all i. If $x_i \ne 0$, then at some point in the execution, the vertex i was chosen at the step $r_j > \tau d_j$. In that iteration, we set $r_i = \tau d_i$. If any other step increments r_i, we must revisit this step and set $r_i = \tau d_i$ again. Then at a solution, $x_i \ne 0$ requires $r_i = \tau d_i$. For such a component, $s_i = 0$, using the definition above. For $x_i = 0$, the value of s_i is irrelevant, and thus, we have $\mathbf{x}^T(\tau\mathbf{d} - \mathbf{r}) = 0$.

This proof makes the nature of ρ immediately clear. If $\rho < 1$, then the output from ACL is *not* equivalent to the solution of Prob. (6). That is, the renormalized solution will *not* satisfy $\mathbf{z}_G^T\mathbf{s} = 0$. Setting $\rho < 1$, however, will compute a solution much more rapidly. This leaves open the question of the precise form of the anti-derivative when $\rho < 1$. We believe that a precise characterization of these solutions exists, but have not been able to obtain a simple form.

4. Illustrative empirical results

In this section, we present several empirical results that illustrate our theory from Section 3. We begin with a few pedagogical examples in order to state solutions of these problems that are correctly normalized. These precise values are sensitive to small differences and imprecisions in solving the equations. We state them here so that others can verify the correctness of their future implementations—although the codes underlying our computations are also available for download.[4] We then illustrate how the 2-norm PageRank vector (Prob. (4)) and 1-norm regularized vector (Prob. (6)) approximate the 1-norm min-cut problem (Prob. (2)) for a small but widely-studied social graph.

4.1. Pedagogical examples

We start by solving numerically the various formulations and variants (min-cut, PageRank, and ACL) of our basic problem for the example graph illustrated in Figure 1. A summary of these results may be found in Table 1. To orient the reader about the normalizations (which can be tricky), we present the three PageRank vectors that satisfy the relationships $\mathbf{x}_{\text{pr}} = \mathbf{D}\mathbf{z}$ and $\text{vol}(S)\mathbf{z} = \mathbf{x}(\alpha, S)$, as predicted by the theory from Section 2 and Theorem 1. Note that $\mathbf{z}$, $\mathbf{x}(\alpha, S)$ and $\mathbf{z}_G$ round to the discrete solution produced by the exact cut if we simply threshold at about half the largest value. Thus, and not surprisingly, all these formulations reveal—to a first approximation—similar properties of the graph. Importantly, though, note also that the vector $\mathbf{z}_G$ has the same sparsity as the true cut-vector. This is a direct consequence of the implicit ℓ_1 regularization that we characterize in Theorem 3. Understanding these "details" matters critically as we proceed to apply these methods to larger graphs where there may be many small entries in the PageRank vector $\mathbf{z}$.[5]

4.2. Newman's netscience graph

Consider, for instance, Newman's netscience graph (Newman, 2006), which has 379 nodes and 914 undirected edges. We considered a set S of 16 vertices, illustrated in Figure 2. In this case, then the solution of the 1-norm min-cut problem (Prob. (2)) has 15 non-zeros; the solution of the PageRank problem (which we have shown in Prob. (4) implicitly solves an ℓ_2 regression problem) has 284 non-zeros;[6] and the solution from the ACL procedure (which we have shown in Prob. (6) solves an ℓ_1-regularized ℓ_2 regression problem) has 24 non-zeros. The true "min-cut" set is large in both the 2-norm PageRank problem and the regularized problem. Thus, we identify the underlying graph feature correctly; but the implicitly regularized ACL procedure does so with many fewer non-zeros than the vanilla PageRank procedure.

Table 1. Numerical results of solving each problem (Prob. (2), Prob. (4), and Prob. (6)) for the graph G and set S from Figure 1. Here, we set $\alpha = 1/2$, $\tau = 10^{-2}$, $\rho = 1$, $\beta = 2/3$, and $\kappa = 0.315$. The vector $\mathbf{x}_{\text{pr}}$, $\mathbf{z}$, and $\mathbf{x}(\alpha, S)$ are the PageRank vectors from Theorem 1, where $\mathbf{x}(\alpha, S)$ solves Prob. (4) and the others are from the problems at the end of Section 2. The vector $\mathbf{x}_{\text{cut}}$ solves the cut Prob. (2), and $\mathbf{z}_G$ solves Prob. (6).

Deg.	$\mathbf{x}_{\text{pr}}$	$\mathbf{z}$	$\mathbf{x}(\alpha, S)$	$\mathbf{x}_{\text{cut}}$	$\mathbf{z}_G$
2	0.0788	0.0394	0.8276	1	0.2758
4	0.1475	0.0369	0.7742	1	0.2437
7	0.2362	0.0337	0.7086	1	0.2138
4	0.1435	0.0359	0.7533	1	0.2325
4	0.1297	0.0324	0.6812	1	0.1977
7	0.1186	0.0169	0.3557	0	0
3	0.0385	0.0128	0.2693	0	0
2	0.0167	0.0083	0.1749	0	0
4	0.0487	0.0122	0.2554	0	0
3	0.0419	0.0140	0.2933	0	0

5. Discussion and Conclusion

We have shown that the PageRank linear system corresponds to a 2-norm variation on a 1-norm formulation of a min-cut problem, and we have also shown that the ACL procedure for computing an approximate personalized PageRank vector *exactly* solves a 1-norm regularized version of the PageRank 2-norm objective. Both of these results are examples of algorithmic anti-differentiation, which involves extracting a precise optimality characterization for the output of an approximate or heuristic algorithmic procedure.

While a great deal of work has focused on deriving new theoretical bounds on the objective function quality or runtime of an approximation algorithm, our focus is somewhat different: we have been interested in understanding what are the precise problems that approximation algorithms and heuristics solve exactly. Prior work has exploited these ideas less precisely in a much larger-scale application to draw very strong downstream conclusions (Leskovec et al., 2009), and our empirical results have illustrated this more precisely in much smaller-scale and more-controlled applications.

Our hope is that one can use these insights in order to combine simple heuristic/algorithmic primitives in ways that will give rise to principled scalable machine learning algorithms more generally. We have early results along these lines that involve semi-supervised learning.

[4] `http://www.cs.purdue.edu/homes/dgleich/codes/l1pagerank`

[5] The reason is both statistical, in that we encourage sparsity to find sparse partitions when the original set S is small, as well as algorithmic, in that the algorithm does *not* in general touch most of the nodes of a large graph in order to find small clusters (Mahoney, 2012). These properties were critically exploited in (Leskovec et al., 2009).

[6] The PageRank solution actually has 379 non-zero entries. However, only 284 of them are non-zero above a 10^{-5} threshold.

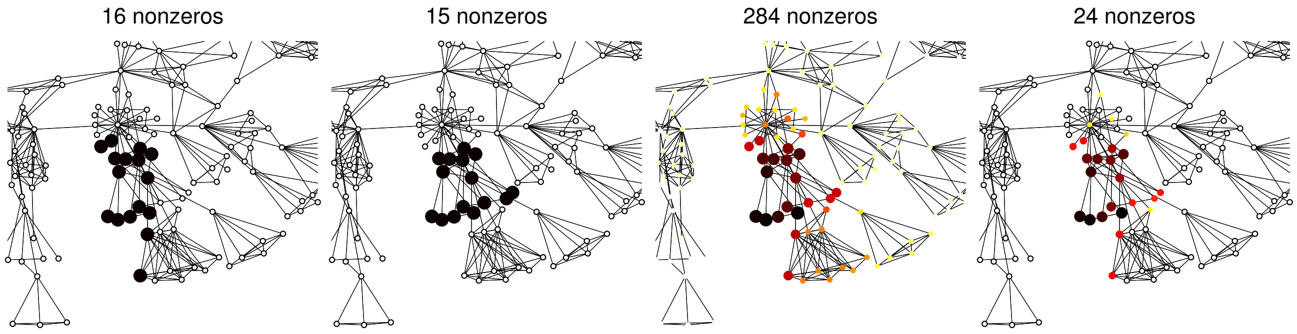

Figure 2. Examples of the different cut vectors on a portion of the netscience graph. In the left subfigure, we show the set S highlighted with its vertices enlarged. In the other subfigures, we show the solution vectors from the various cut problems (from left to right, Probs. (2), (4), and (6), solved with min-cut, PageRank, and ACL) for this set S. Each vector determines the color and size of a vertex, where high values are large and dark. White vertices with outlines are numerically non-zero (which is why most of the vertices in the fourth figure are outlined, in contrast to the third figure). The true min-cut set is large in all vectors, but the implicitly regularized problem achieves this with many fewer non-zeros than the vanilla PageRank problem.

References

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

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*, pp. 475–486, 2006.

Arasu, Arvind, Novak, Jasmine, Tomkins, Andrew, and Tomlin, John. PageRank computation and the structure of the web: Experiments and algorithms. In *Proceedings of the 11th international conference on the World Wide Web*, 2002.

Chin, Hui Han, Mądry, Aleksander, Miller, Gary L., and Peng, Richard. Runtime guarantees for regression problems. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, pp. 269–282, 2013.

Dhillon, Inderjit S., Guan, Yuqiang, and Kulis, Brian. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.

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

Koutra, Danai, Ke, Tai-You, Kang, U., Chau, Duen Horng, Pao, Hsing-Kuo Kenneth, and Faloutsos, Christos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases*, pp. 245–260, 2011.

Kulis, Brian and Jordan, Michael I. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Langville, Amy N. and Meyer, Carl D. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

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, 2009.

Mahoney, Michael W. Approximate computation and implicit regularization for very large-scale data analysis. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pp. 143–154, 2012.

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.

Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, 2006.

Orecchia, Lorenzo and Mahoney, Michael W. Implementing regularization implicitly via approximate eigenvector computation. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 121–128, 2011.

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

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, November 1999.

Saunders, Michael A. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT Numerical Mathematics*, 35(4): 588–604, 1995.

Tibshirani, Robert. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58: 267–288, 1994.

Zhou, Dengyong, Bousquet, Olivier, Lal, Thomas Navin, Weston, Jason, and Schölkopf, Bernhard. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pp. 321–328, 2004.