Exploiting Optimization for Local Graph Clustering

Kimon Fountoulakis* Xiang Cheng† Julian Shun‡ Farbod Roosta-Khorasani§
Michael. W. Mahoney¶

February 4, 2016

Abstract

Local graph clustering methods aim to identify well-connected clusters around a given "seed set" of reference nodes. The main focus of prior theoretical work has been on worst-case running time properties or on implicit statistical regularization; and the focus of prior empirical work has been to identify structure in large social and information networks. Here, we adopt an optimization perspective on local graph clustering methods. In particular, we clarify the relationship between the local spectral algorithm of [3] and a variant of a well-studied optimization objective. This insight permits us to develop a local spectral graph clustering algorithm that has improved theoretical convergence properties. We also demonstrate the numerical performance of this optimization-based algorithm and some heuristic variants of it.

Keywords. local graph clustering; Page-Rank; ℓ_1-regularization; large scale optimization; coordinate descent; iteration complexity.

AMS Classification. 49M37; 65K05; 90C06; 90C25; 90C35; 90C27; 91C20; 05C50.

1 Local Graph Clustering

Modern graph clustering applications require the analysis of large graphs [14, 17] and in many cases the large size of recent graph data has rendered classical approaches [5, 12, 13, 16, 23] computationally too expensive. The reason is that the running time typically increases with the size of the input graph. This problem sparked the development of methods [1, 2, 3, 8, 9, 15, 21, 25] that are *local*, in that their running time depends on the size of the output or on the size of an input seed set of reference nodes (and not the size of the full graph). Local graph clustering methods have been mainly motivated by the need for faster algorithms on large-scale graphs. A second motivation is that many real-world graphs tend to have "good" local and small/medium size clusters, as opposed to "good" large ones [14, 17].

The goal of this paper is to interpret one such algorithm, i.e., that of [3], from an optimization perspective, thereby enabling us more easily to use ideas from optimization in a more black box manner. This new interpretation will give insight on the performance of existing methods and will allow the development of new methods.

Before we proceed to describe our main contributions, let us state here what we mean by local versus global algorithms. By *global algorithms*, we refer to algorithms with running time that depends on the number of nodes and/or the number of edges of the given graph. By *local (or strongly-local) algorithms*, we refer to

*K. Fountoulakis is with the International Computer Science Institute, Department of Statistics, University of California Berkeley, 1947 Center St, Ste. 600, Berkeley, CA 94704, USA. e-mail: kfount@berkeley.edu.

†X. Cheng is with the Department of Electrical Engineering and Computer Science, University of California Berkeley, 253 Cory Hall, Berkeley, CA 94720-1770, USA. e-mail: x.cheng@berkeley.edu.

‡J. Shun is with the Department of Electrical Engineering and Computer Science, Department of Statistics, University of California Berkeley, 253 Cory Hall, Berkeley, CA 94720-1770, USA. e-mail: jshun@eecs.berkeley.edu.

§F. Roosta-Khorasani is with the International Computer Science Institute, Department of Statistics, University of California Berkeley, 1947 Center St, Ste. 600, Berkeley, CA 94704, USA. e-mail: farbod@icsi.berkeley.edu.

¶M. Mahoney is with the International Computer Science Institute, Department of Statistics, University of California Berkeley, Evans Hall, 2594 Hearst Ave., Berkeley, CA 94720, USA. e-mail: mmahoney@stat.berkeley.edu.

algorithms with running time that depends on the number of nodes and/or edges in the input seed set or output cluster. Finally, there exists a class of *locally-biased algorithms*, e.g., that of [18], where the running time depends on the whole graph, but where the solution is locally-biased toward some input seed set of reference nodes. In this paper, we are interested in algorithms that satisfy the first two definitions, but doing so will require us to understand precisely the relationship between strongly-local and locally-biased algorithms from an optimization perspective.

2 Summary of Main Contributions

Our main results from the optimization perspective are based on the fact that we can relate the Approximate Page-Rank (APR) algorithm in [3] to a coordinate descent algorithm for a quadratic objective function.

- *Termination condition:* Based on [11], we make explicit why the optimality conditions of an ℓ_1-regularized Page-Rank (PR) problem imply the special termination criterion of APR, and thus provide same Cheeger-like guarantees as the ones presented in [3].
- *Fixing the output solution:* In comparison to APR, which combines both properties of local/sparse solutions and local algorithms, the proposed ℓ_1-reg. PR problem decouples the locality/sparsity of the solution from properties of the algorithm, i.e., which nodes are used at every iteration. If there exists a local solution, then any algorithm that solves the proposed ℓ_1-reg. PR problem will obtain the same local solution (while perhaps running in time that depends on the size of the entire graph).
- *Optimization algorithms:* We present an algorithm based on iterative soft-thresholding algorithm (ISTA) [7] that solves the ℓ_1-reg. PR problem, while maintaining a running time of order of the number of nodes/non-zeros in the optimal solution (i.e., independent of the size of the graph). We also present a block-coordinate descent algorithm that is more flexible to heuristic modifications and thus can be made fast in practice.
- *Linear convergence rate:* Let $\rho > 0$ be the accuracy parameter in [3]. We show that ISTA has *linear* convergence rate, i.e., the iteration complexity is $\mathcal{O}(\log(1/\rho))$, while achieving running time that depends on the size of the output set of nodes. On the contrary, APR is currently known to have sub-linear convergence rate, i.e., iteration complexity $\mathcal{O}(1/\rho)$.
- *Implementations:* We implement carefully in C++ the proposed algorithms such that they have strongly local running time. We illustrate numerical experiments on several large-scale graphs, which demonstrate the above claims and show that the proposed algorithms are competitive with APR in [3].

3 Preliminaries and Notation

We will denote the i'th coordinate of a vector p with $p(i)$. The vector e denotes the vector of all ones and the vector e_i denotes a vector which has one at the i'th component and zero elsewhere. We overload the square root operator for vectors q, i.e., $q^{1/2} := [q_1^{1/2}, \ldots, q_n^{1/2}]$.

We assume that we are given an unweighted and undirected graph $\mathcal{G}$ with no self-loops. We denote with $\mathcal{V}$ the set of nodes of graph $\mathcal{G}$. The number of nodes and edges of $\mathcal{G}$ are denoted by n and m, respectively. We define $[n] := \{1, 2, \ldots, n\}$. We denote by $j \sim i$ that j is a neighbor of i. For a set of nodes S, let $j \sim S$ denote a node j that is a neighbor of at least one node in S. Let A be the adjacency matrix of $\mathcal{G}$, D the diagonal degree matrix of $\mathcal{G}$, $Q := D^{-1/2}\left\{D - \frac{1-\alpha}{2}(D+A)\right\}D^{-1/2}$ and $f(q) := \frac{1}{2}q^TQq - \alpha s^TD^{-1/2}q$. Let $S \subseteq [n]$ and $I_S \in \mathbb{R}^{n\times|S|}$ be a selection of columns of the identity matrix with indices in S. Further, we define $\nabla_S f(q) := I_S^T\nabla f(q)$, $d_S := diag(I_S^TDI_S)$ and $Q_S := I_S^TQI_S$, where $diag(\dot{})$ extracts the diagonal of the input matrix and returns it as a vector. Let $\text{vol}(S) := \sum_{i\in S} d_i$ be the volume of $S \subseteq \mathcal{V}$ and $\text{supp}(q) := \{i \in [n] \mid q_i \neq 0\}$. The function ∇f is block 1-Lipschitz continuous and the function f is α-strongly-convex, both w.r.t. ℓ_2 norm.

Let us define $\|q\|_{\infty,j} = \{\text{jth largest } |q(j)|\}$ and

$$\|x\|_{(\tau)} = (\sum_{j=1}^{\tau} \|x\|_{\infty,j}^2)^{1/2},$$

where τ is a positive integer. Notice that $\|\cdot\|_{(1)} = \|\cdot\|_\infty$ and $\|\cdot\|_{(n)} = \|\cdot\|_2$. It is shown in Section 2 in [4] that this is a norm and in Proposition 3.1 in [4] that the dual norm $\|x\|_{(\tau),*}$ of $\|x\|_{(\tau)}$ satisfies $\|x\|_{(\tau),*} \leq$

$\sqrt{2}\max\{\|x\|_2, (1/\sqrt{\tau})\|x\|_1\}$. Moreover, using the definition of $\|x\|_{(\tau),*}$ in Proposition 2.1 in [4], we have that $\|\cdot\|_{(1),*} = \|\cdot\|_1$ and $\|\cdot\|_{(n),*} = \|\cdot\|_2$. Let S be a set of nodes and $\tau \leq |S|$, we define

$$\mu_\tau^S := \min_{u\in\mathbb{R}^{|S|}, \|u\|_{(\tau),*}=1} \langle u, Q_S u\rangle.$$

It is easy to see that $\mu_\tau^S \geq (1/2)\min(\mu_{|S|}^S, \tau\mu_1^S)$. Note that $\mu_{|S|}^S \geq \alpha$, where α is a lower bound on the smallest eigenvalue of matrix Q. From Theorem 2.1.9 in [19] we have that the function f is $\mu_\tau^{[n]}$-strongly-convex w.r.t. the $\|x\|_{(\tau),*}$ norm.

4 Approximate Page-Rank in a Nutshell

APR solves approximately the Page-Rank linear system

$$p = \alpha s + (1-\alpha)Wp, \tag{1}$$

where p is the unknown vector and $W = (I + AD^{-1})/2$ is a lazy random walk matrix. Let us define the residual as $r := (I - (1-\alpha)W)p - \alpha s$. A coordinate solver applied to (1) updates the current approximate solution according to $p_{k+1} = p_k - r_k(i)e_i$. Then the residual is updated as

$$r_{k+1} = r_k - r_k(i)e_i + \frac{1-\alpha}{2}(I + AD^{-1})r_k(i)e_i.$$

Using the above and the definitions of D and A we obtain the new residual r_{k+1} in Steps 5, 6 and 7 of Algorithm 1. By defining $\tilde{r}_k := -(1/\alpha)r_k$ and replacing r_k with $\tilde{r}_k$ in Algorithm 1 we obtain APR in the same form as

Algorithm 1 Coordinate solver for (1)

1: **Initialize**: $\rho > 0$, $p_0 = 0$, thus $r_0 = -\alpha s$
2: **while** $\|D^{-1}r_k\|_\infty > \rho\alpha$ **do**
3: Choose an i such that $r_k(i) < -\alpha d_i \rho$
4: $p_{k+1}(i) = p_k(i) - r_k(i)$
5: $r_{k+1}(i) = \frac{1-\alpha}{2} r_k(i)$
6: For each j such that $j \sim i$ set
$$r_{k+1}(j) = r_k(j) + \frac{1-\alpha}{2d_i} r_k(i)$$
7: For each j such that $j \nsim i$ set $r_{k+1}(j) = r_k(j)$
8: $k = k + 1$
9: **end while**
10: **return** p_k

described in [3].

To show that Algorithm 1 is equivalent to a coordinate descent algorithm we need to define an optimization problem. It is easy to show that Algorithm 1 solves the optimization problem "minimize"$f(q)$, where f is defined in Section 3. To see this, note that the residual in Algorithm 1 can be written in terms of the scaled gradient of the function f. In particular, the gradient of the previous objective function is $\nabla f(q) = D^{-1/2}\left\{D - \frac{1-\alpha}{2}(D+A)\right\}D^{-1/2}q - \alpha D^{-1/2}s$ and thus $D^{1/2}\nabla f(q) = r$, where $q := D^{-1/2}p$. Using $D^{1/2}\nabla f(q) = r$ we can rewrite Algorithm 1 as a coordinate descent method in Algorithm 2 for minimizing the function f.

5 ℓ_1-Regularized Page-Rank

Algorithm 2 terminates when

$$\|D^{-1/2}\nabla f(q_k)\|_\infty \leq \rho\alpha. \tag{2}$$

In the following lemma it is shown that Algorithm 2 satisfies $\nabla f(q_k) \leq 0\ \forall k$.

Algorithm 2 Coordinate descent solver

Initialize: $\rho > 0$, $q_0 = 0$, thus $\nabla f(q_0) = -\alpha D^{-1/2} s$
while $\|D^{-1/2}\nabla f(q_k)\|_\infty > \rho\alpha$ **do**
  Choose an i such that $\nabla_i f(q_k) < -\alpha\rho d_i^{1/2}$
  $q_{k+1}(i) = q_k(i) - \nabla_i f(q_k)$
  $\nabla_i f(q_{k+1}) = \frac{1-\alpha}{2}\nabla_i f(q_k)$
  For each j such that $j \sim i$ set

$$\nabla_j f(q_{k+1}) = \nabla_j f(q_k) + \frac{(1-\alpha)}{2d_i^{1/2}d_j^{1/2}}\nabla_i f(q_k)$$

  For each j that $j \nsim i$ set $\nabla_j f(q_{k+1}) = \nabla_j f(q_k)$
  $k = k + 1$
end while
return $p_k := D^{1/2} q_k$

Lemma 1. *If Algorithm 2 is initialized with $q_0 = 0$ and $s \geq 0$, then $q_{k+1} \geq q_k$ and $\nabla f(q_k) \leq 0\ \forall k$.*

Using Lemma 1, since the gradient components at every iteration are all non-positive we have that the termination criterion of Algorithm 2 is equivalent to

$$\nabla_i f(q_k) \geq -\rho\alpha d_i^{1/2}\ \forall i. \tag{3}$$

The termination criterion (3) of Algorithm 2 is related to the first-order optimality conditions of the following ℓ_1-regularized problem

$$\text{minimize} \quad \psi(q) := \rho\alpha\|D^{1/2}q\|_1 + f(q). \tag{4}$$

Let us denote the optimal solution of problem (4) with q_*. The first-order optimality conditions of (4) are

$$\nabla_i f(q_*) = \begin{cases} -\rho\alpha d_i^{1/2} & \text{if } q_*(i) > 0 \\ \rho\alpha d_i^{1/2} & \text{if } q_*(i) < 0 \\ \in \rho\alpha d_i^{1/2}[-1, 1] & \text{if } q_*(i) = 0. \end{cases}$$

In Theorem 1 we will prove that the solution of problem (4) satisfies $q_* \geq 0$. Therefore, the optimality conditions of problem (4) are equivalent to

$$\nabla_i f(q_*) = \begin{cases} -\rho\alpha d_i^{1/2} & \text{if } q_*(i) > 0 \\ \in \rho\alpha d_i^{1/2}[-1, 0] & \text{if } q_*(i) = 0. \end{cases} \tag{5}$$

Notice that the optimality conditions (5) imply the termination criterion (3) of Algorithm 2, but the converse is not necessarily true. This is because the termination criterion of Algorithm 2 does not distinguish between positive and zero components of q_*.

It is worth mentioning that the above results are motivated by Theorem 3 in [11]. However, we make clear the connection of the termination criterion of Algorithm 2 and the first-order optimality conditions of (4). This allows us to get a much simpler form of the ℓ_1-reg. PR problem than the form that is presented in [11]. In addition we show in Section 8 that by solving problem (4), one can theoretically achieve the same worst-case local graph clustering guarantees as APR.

It is important to make a comment on the number of non-zeros/nodes in the optimal solution of the ℓ_1-reg. PR problem and the output of APR. Based on Theorem 1 in [3] the output of APR has $\mathcal{O}(1/(\rho\alpha))$ non-zeros at termination in the worst-case. It is known [6, 26] that ℓ_1 regularization promotes sparsity in the optimal solution, but it is difficult to have certain guarantees on the number of non-zeros unless we make assumptions about possible models of the given graph. This will result in interesting statistical goals, i.e., that of recovering assumed sparse models over the nodes of the graph [27]. This is not our objective in this paper, however, we do provide numerical evidence in Section 9 showing that the sparsity pattern of the output of APR and the optimal solution of the ℓ_1-reg. PR problem are very similar.

6 ISTA for ℓ_1-Regularized Page-Rank

The main computational advantage of coordinate descent APR is the *locality* of its iterations. More specifically, throughout the algorithm, the only coordinates updated are the ones which are nonzero in the final solution. This makes application of APR very appealing for modern large graphs. We present a new algorithm, Algorithm 3, for solving ℓ_1-reg. PR (4), where the updates are chosen carefully to maintain this *locality* property while also guaranteeing linear convergence.

Algorithm 3 ISTA-based solver for (4)

1: **Initialize**: $\epsilon \in (0,1)$, $\alpha > 0$, $q_0 = 0$, thus $\nabla f(q_0) = -\alpha D^{-1/2} s$. Moreover, choose ρ such that $\|D^{-1/2}\nabla f(q_0)\|_\infty > \rho\alpha$.
2: **while** $\|D^{-1/2}\nabla f(q_k)\|_\infty > (1+\epsilon)\rho\alpha$ **do**
3: Set $S_k := \{i \in [n] \mid q_k(i) - \nabla_i f(q_k) \geq \rho\alpha d_i^{1/2}\}$
4: $(p_{k+1}, \nabla f(q_{k+1})) \leftarrow$**Step**$(q_k, \nabla f(q_k), S_k)$
5: $k = k+1$
6: **end while**
7: **return** $p_k := D^{1/2} q_k$

Step$(q_k, \nabla f(q_k), S)$

1: $\Delta q_k := -(\nabla_S f(q_k) + \rho\alpha d_S^{1/2})$ and
$q_{k+1}(S) = q_k(S) + \Delta q_k$
2: For each $i \in S$ set

$$\nabla_i f(q_{k+1}) = -\rho\alpha d_i^{1/2} - \frac{1-\alpha}{2}[I_S \Delta q_k]_i - \frac{1-\alpha}{2d_i^{1/2}} \sum_{l \sim i, l \in S} \frac{A_{i,l}[I_S \Delta q_k]_l}{d_l^{1/2}})$$

3: For each $j \notin S$ such that $j \sim S$ set

$$\nabla_j f(q_{k+1}) = \nabla_j f(q_k) - \frac{1-\alpha}{2d_j^{1/2}} \sum_{l \sim j, l \in S} \frac{A_{j,l}[I_S \Delta q_k]_l}{d_l^{1/2}}$$

4: For each $j \notin S$ such that $j \nsim S$ set

$$\nabla_j f(q_{k+1}) = \nabla_j f(q_k)$$

5: Return: q_{k+1} and $\nabla f(q_{k+1})$

The size of the set S_k in Algorithm 3 changes at every iteration. In Theorem 1, we show that $|S_k|$ is bounded by the number of non-zeros of the optimal solution $\forall k$. As a result, one can update all the coordinates in set S_k and still have a per-iteration cost that depends only on the sparsity of the solution, as opposed to the size of the full graph. Furthermore, we show in Theorem 1 that Algorithm 3 is indeed equivalent to a standard *proximal full gradient descent* algorithm as described in [7], which is also known as iterative soft-thresholding algorithm (ISTA).

Theorem 1. *Algorithm 3 is equivalent to ISTA in [7], which is a proximal full gradient descent algorithm. Let q_* be the optimal solution of* (4)*. Then we have that $|S_k| \leq |supp(q_*)|$ and $|S_k| \leq |S_{k+1}|$ $\forall k$. Algorithm 3 satisfies $q_k \geq 0$ $\forall k$, which implies that the optimal solution of problem* (4) *is positive. Algorithm 3 satisfies $\nabla f(q_k) \leq 0$ and $q_{k+1} \geq q_k$ $\forall k$. Moreover, we have that $\nabla_i f(q_k) \leq -\rho\alpha d_i^{1/2}$ $\forall i \in S_k$ and $\nabla_i f(q_k) > -\rho\alpha d_i^{1/2}$ $\forall i \in [n] \backslash S_k$ $\forall k$.*

The proof is given in the appendix.

Block-ISTA: We present a block coordinate algorithm, shown in Algorithm 4. Our main motivation for proposing this algorithm is its good practical performance, which we present in Section 9. In situations where

the set S_k grows rapidly, one might prefer to allow for only a small fraction of number of coordinates in S_k to be selected at every iteration. Algorithm 3 allows for this flexibility. In fact, a trivial modification in the proof Theorem 1 shows that Algorithm 4 is equivalent to block-ISTA, also known as the *block proximal coordinate descent* algorithm [20, 24], and all the properties mentioned in Theorem 1 hold.

Algorithm 4 Block coordinate descent solver for (4)

1: **Initialize**: $\epsilon \in (0,1)$, $\alpha \in (0,1)$, $\tau_{\max} \in \mathbb{Z}^+$, $\tau_{\min} \in \mathbb{Z}^+ \leq \min(|\text{supp}(s)|, \tau_{\max})$, $q_0 = 0$ thus $\nabla f(q_0) = -\alpha D^{-1/2} s$. Moreover, choose ρ such that $\|D^{-1/2}\nabla f(q_0)\|_\infty > \rho\alpha$.
2: **while** $\|D^{-1/2}\nabla f(q_k)\|_\infty > (1+\epsilon)\rho\alpha$ **do**
3: Set $S_k := \{i \in [n] \mid q_k(i) - \nabla_i f(q_k) \geq \rho\alpha d_i^{1/2}\}$
4: Choose $\tau_k \in \mathbb{Z}^+$ and $S_k^{\tau_k} \subseteq S_k$ such that $S_k^{\tau_k}$ contains the $|S_k^{\tau_k}| = \max(\min(\tau_k, \tau_{\max}, |S_k|), \tau_{\min})$ largest elements in absolute value in S_k.
5: $(p_{k+1}, \nabla f(q_{k+1})) \leftarrow$**Step**$(q_k, \nabla f(q_k), S_k^{\tau_k})$
6: $k = k+1$
7: **end while**
8: **return** $p_k := D^{1/2} q_k$

7 Complexity Analysis

In this section we study worst-case iteration complexity of Algorithms 3 and 4 and their running times.

7.1 Analysis for Algorithm 3

From Theorem 1 we have that $q_k \geq 0\ \forall k$, i.e., we always remain in the the non-negative orthant. Denoting the restriction of $\psi(q)$ to $q \geq 0$, by

$$\widehat{\psi}(q) := \rho\alpha e^T D^{1/2} q + f(q),$$

it follows that $\psi(q) = \hat{\psi}(q)$ for all q in the non-negative orthant. From 1-Lipschitz continuity of ∇f w.r.t. ℓ_2 norm, it follows that $\widehat{\psi}$ is also smooth with the same parameter, i.e., 1. Hence, for any q_k from Algorithm 3, we have

$$\widehat{\psi}(q) \leq \psi(q_k) + (q - q_k)^T \nabla\widehat{\psi}(q_k) + \frac{1}{2}\|q_k - q\|_2^2. \tag{6}$$

Since $q_{k+1} \geq 0$ (see Theorem 1), $q_{k+1} - q_k = I_{S_k}\Delta q_k$ and $\Delta q_k = -\nabla_{S_k}\widehat{\psi}(q_k)$ we have that

$$\psi(q_{k+1}) \leq \psi(q_k) - \frac{1}{2}\|\nabla_{S_k}\widehat{\psi}(q_k)\|_2^2. \tag{7}$$

By α-strong convexity of ψ we have

$$\psi(q_k) - \psi(q_*) \leq \frac{1}{2\alpha}\|g\|_2^2 \quad \forall g \in \partial\psi(q_k),$$

where $\partial\psi(q_k)$ is the sub-differential of ψ at q_k. Notice that $I_{S_k}\Delta q_k$ is a valid sub-gradient of ψ at q_k, and $\|I_{S_k}\Delta q_k\|_2^2 = \|\nabla\widehat{\psi}(q_k)(S_k)\|_2^2$. This gives us

$$\psi(q_k) - \psi(q^*) \leq \frac{1}{2\alpha}\|\nabla_{S_k}\widehat{\psi}(q_k)\|_2^2. \tag{8}$$

Combining (7) and (8) and subtracting $\psi(q_*)$ from both sides we get

$$\psi(q_{k+1}) - \psi(q^*) \leq (1-\alpha)(\psi(q_k) - \psi(q_*)),$$

which implies linear convergence. Applying the last inequality recursively we get that Algorithm 3 requires at most $T \in \mathcal{O}(\log(1/\hat{\epsilon})/\alpha)$ iterations to obtain a solution q_T such that $\psi(q_T) - \psi(q_*) \leq \hat{\epsilon}$.

From (7) we have that

$$\psi(q_*) \leq \psi(q_k) - \frac{1}{2}\|\nabla_{S_k}\widehat{\psi}(q_k)\|_2^2.$$

Using the above and $\psi(q_T) - \psi(q_*) \leq \hat{\epsilon}$, we get $\|\nabla_{S_k}\widehat{\psi}(q_k)\|_\infty^2 \leq 2\hat{\epsilon}$, which is equivalent to

$$\rho\alpha - \left(\frac{2\hat{\epsilon}}{d_i}\right)^{1/2} \leq \frac{\nabla_i f(q_T)}{d_i^{1/2}} \leq -\rho\alpha + \left(\frac{2\hat{\epsilon}}{d_i}\right)^{1/2}$$

$\forall i \in S_k$. From Theorem 1 we have that $\nabla_i f(q_T) > -\rho\alpha d_i^{1/2}\ \forall i \in [n]\backslash S_k$. As a result, by setting $\hat{\epsilon} := (\rho^2\alpha^2 \min_j d_j)/2$ and using the fact that $\nabla f(q_k) \leq 0\ \forall k$ from Lemma 1, we get that after

$$T \in \mathcal{O}\left(\frac{1}{\alpha}\log\left(\frac{2}{\rho^2\alpha^2 \min_j d_j}\right)\right) \tag{9}$$

iterations the output of Algorithm 3 satisfies $-2\rho\alpha d_i^{1/2} \leq \nabla_i f(q_T) \leq 0\ \forall i$. This, in particular, implies that if the original APR Algorithm 2 is initialized with $\tilde{\rho}$ an returns an output that satisfies (3), then setting $\rho := \tilde{\rho}/2$ in Algorithm 3, ensures that both Algorithms 2 and 3 terminate, offering the same guarantees. From (9) we conclude that the iteration complexity of ISTA depends logarithmically on ρ.

Now we discuss the running time of Algorithm 3. Let $\mathcal{S}_* := \text{supp}(q_*)$. From Theorem 1 we have that $|S_k| \leq |\mathcal{S}_*|$. Hence, Step 1 in the **Step** procedure requires at most $\mathcal{O}(|\mathcal{S}_*|)$ operations. Similarly Steps 2 and 3 require at most $\mathcal{O}(|\mathcal{S}_*| + \text{vol}(\mathcal{S}^*))$ operations. This can be upper bounded by $\mathcal{O}(|\mathcal{S}_*|(1+\max_{i\in\mathcal{S}^*} d_i))$. Finally, Step 4 does not perform any computations. Putting the operations performed in all of the steps together and using the iteration complexity result in (9) we have that Algorithm 3 requires at most

$$\mathcal{O}\left(\frac{|\mathcal{S}_*|(1+\max_{j\in\mathcal{S}^*} d_j)}{\alpha}\log\left(\frac{2}{\rho^2\alpha^2 \min_j d_j}\right)\right)$$

time.

7.2 Analysis for Algorithm 4

The analysis in this section follows the analysis in Section 4 in [20] but with many modifications to adapt to our specific problem. We also analyze block coordinate descent instead of single coordinate descent. From Step 4 of Algorithm 4 we have that at every iteration $\tilde{\tau}_k := |S_k^{\tau_k}| = \max(\min(\tau_k, \tau_{\max}, |S_k|), \tau_{\min})$ coordinates are updated. From Step 5 of Algorithm 4, we see that $\Delta q_k = -\nabla_{S_k^{\tau_k}}\widehat{\psi}(q_k)$, hence using the block 1-Lipschitz continuity of $\nabla\widehat{\psi}$ w.r.t. the ℓ_2-norm and non-negativity of q_k and $q_{k+1} = q_k + I_{S_k^{\tau_k}}\Delta q_k$ (see Theorem 1 which holds for Algorithm 4 as well), we get that at iteration k

$$\begin{aligned}\psi(q_{k+1}) = \widehat{\psi}(q_{k+1}) &\leq \widehat{\psi}(q_k) - \frac{1}{2}\|\nabla_{S_k^{\tau_k}}\widehat{\psi}(q_k)\|_{(\tilde{\tau}_k)}^2. \\ &= \psi(q_k) - \frac{1}{2}\|\Delta q_k\|_{(\tilde{\tau}_k)}^2. \end{aligned} \tag{10}$$

Let $\mathcal{S}_* := \text{supp}(q_*)$. From Theorem 1 we have that q_k and q_* are supported in $\mathcal{S}_*$. Since f is strongly-convex w.r.t. $\|\cdot\|_{(\tilde{\tau}_k),*}$ so is ψ. Using $\mathcal{S}_*$ we can redefine strong convexity

$$\psi(I_{\mathcal{S}_*}p) \geq \psi(I_{\mathcal{S}_*}q) + (p-q)^T I_{\mathcal{S}_*}^T g + \frac{\mu_{\tilde{\tau}_k}^{\mathcal{S}_*}}{2}\|I_{\mathcal{S}_*}(p-q)\|_{(\tilde{\tau}_k),*}^2,$$

where $g \in \partial\psi(I_{\mathcal{S}_*}q)$. Minimizing both sides w.r.t. $p \in \mathbb{R}^{\mathcal{S}_*}$ and using conjugacy we get

$$\begin{aligned}\psi(q_*) &\geq \psi(I_{\mathcal{S}_*}q) - \left(\frac{\mu_{\tilde{\tau}_k}^{\mathcal{S}_*}}{2}\|\cdot\|_{(\tilde{\tau}_k),*}^2\right)^*(-I_{\mathcal{S}_*}g) \\ &= \psi(I_{\mathcal{S}_*}q) - \frac{1}{2\mu_{\tilde{\tau}_k}^{\mathcal{S}_*}}\|I_{\mathcal{S}_*}g\|_{(\tilde{\tau}_k)}^2.\end{aligned}$$

Let us define $\Delta\tilde{q}_k = I_{\mathcal{S}_*}^T I_{S_k^{\tau_k}} \Delta q_k$. We can replace $I_{\mathcal{S}_*} q$ with $I_{\mathcal{S}_*} I_{\mathcal{S}_*}^T q_k = q_k$ and using the fact that $I_{\mathcal{S}_*} \Delta\tilde{q}_k = I_{S_k^{\tau_k}} \Delta q_k$ is the sub-gradient of ψ at q_k, we get that

$$\psi(q_*) \geq \psi(q_k) - \frac{1}{2\mu_{\tilde{\tau}_k}^{\mathcal{S}_*}} \|\Delta q_k\|_{(\tilde{\tau}_k)}^2. \tag{11}$$

Since Δq_k is zero only if $q_k = q_*$ then (10) guarantees that Algorithm 4 converges asymptotically to the optimal solution of the ℓ_1-reg. PR problem. However, to obtain a non asymptotic worst-case iteration complexity result we need to lower bound $\tilde{\tau}_k$. From the definition of $\tilde{\tau}_k$ we have that $\tilde{\tau}_k \geq \tau_{\min}\ \forall k$. From Section 3 we have $\mu_{\tilde{\tau}_k}^{\mathcal{S}_*} \geq 1/2 \min(\mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}, \tilde{\tau}_k \mu_1^{\mathcal{S}_*})$. Using $\tilde{\tau}_k \geq \tau_{\min}\ \forall k$ we get $\mu_{\tilde{\tau}_k}^{\mathcal{S}_*} \geq 1/2 \min(\mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}, \tau_{\min} \mu_1^{\mathcal{S}_*})$. Combining (10) and (11), using the latter inequality and subtracting $\psi(q_*)$ from both sides we get

$$\psi(q_{k+1}) - \psi^* \leq \left(1 - \frac{\min(\mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}, \tau_{\min}\mu_1^{\mathcal{S}_*})}{2}\right)(\psi(q_k) - \psi^*). \tag{12}$$

Using the above inequality recursively we get that Algorithm 4 requires at least

$$T \in \mathcal{O}\left(\frac{1}{\min(\mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}, \tau_{\min}\mu_1^{\mathcal{S}_*})} \log\left(\frac{1}{\hat{\epsilon}}\right)\right)$$

iterations to obtain a solution q_T that satisfies $\psi(q_T) - \psi^* \leq \hat{\epsilon}$. Similarly to Algorithm 3, by setting $\hat{\epsilon} := (\rho^2 \alpha^2 \min_j d_j)/2$ and using $\rho/2$ instead of ρ then Algorithm 4 has the same guarantees as APR, i.e., $-\rho\alpha d_i^{1/2} \leq \nabla_i f(q_T) \leq 0\ \forall i$. It is important to mention that $|\mathcal{S}^*|$ depends on ρ, therefore, the iteration complexity of block-ISTA does not have logarithmic dependence on ρ. If we consider $\hat{\epsilon}$ a constant independent of ρ, then we get logarithmic dependence of the iterations on $\hat{\epsilon}$, but then block-ISTA has approximately the same guarantees as ISTA. Regarding the running time of block-ISTA we have that at each iteration of Algorithm 4 at most $\tau_{\max}$ coordinates plus their neighbors are used. Therefore, each iteration of the algorithm requires in worst-case $\mathcal{O}(\tau_{\max}(1+\max_{i\in\mathcal{S}^*} d_i))$ running time. Putting everything together we have that the running time of Algorithm 4 is

$$\mathcal{O}\left(\frac{\tau_{\max}(1 + \max_{i\in\mathcal{S}^*} d_i)}{\min(\mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}, \tau_{\min}\mu_1^{\mathcal{S}_*})} \log\left(\frac{2}{\rho^2\alpha^2 \min_j d_j}\right)\right).$$

Regarding the strong convexity parameter $\mu_1^{\mathcal{S}_*}$ it is shown in Section 4 in [20] that $\mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}/|\mathcal{S}_*| \leq \mu_1^{\mathcal{S}_*} \leq \mu_{|\mathcal{S}_*|}^{\mathcal{S}_*}$. Due to this lower bound of $\mu_1^{\mathcal{S}_*}$ we cannot conclude that block-ISTA has better running time than ISTA.

7.3 Comparison to APR

The analysis of APR in [3], suggests that its iteration complexity is $\mathcal{O}(1/\rho)$, which corresponds to a sub-linear convergence rate. We showed in Section 7 that ISTA achieves an iteration complexity $\mathcal{O}(\log(1/\rho))$, which corresponds to a linear rate.

Another important aspect is the dependence of the running time of the algorithms on the number of non-zeros of the solution. The worst-case running time of Algorithms 3 and 4 is $\mathcal{O}(|\mathcal{S}_*|(1+\max_{i\in\mathcal{S}^*} d_i))$; ignoring small terms. Theorems 1 and 5 in [3] state that the iteration complexity of APR depends on the volume of the cluster that APR is initialized to. Let us denote this cluster with $\mathcal{S}_{APR}$, then we have that $\text{vol}(\mathcal{S}_{APR}) \leq |\mathcal{S}_{APR}| \max_{i\in\mathcal{S}_{APR}} d_i$; notice that we used a similar bound for the analysis of the running time of ISTA and block-ISTA. One iteration of APR costs $\mathcal{O}(1+\max_{i\in\mathcal{S}_{APR}} d_i)$ in worst-case. Therefore, the worst-case running time of APR $\mathcal{O}(|\mathcal{S}_{APR}|(\max_{i\in\mathcal{S}_{APR}} d_i)^2)$. Note that we can improve on the bound given in APR by choosing greedily the coordinate with the largest partial derivative in absolute value at each iteration and using the same convex analysis tools that we used for Algorithm 4. By doing so, we can obtain a running time of APR that is $\mathcal{O}(|\mathcal{S}_{APR}|(1 + \max_{i\in\mathcal{S}_{APR}} d_i))$. Therefore the only difference in the running time bounds between APR and ISTA is the term denoting the number of non-zeros in their solutions. While we cannot prove anything about the difference in solution sizes between the two algorithms, we provide empirical evidence in Section 9 showing that the number of non-zeros of Algorithms 3 and 4 is approximately the number of non-zeros for the output of APR.

8 Using APR to find low-conductance cuts

The main application of APR in [3] is to approximately solve the NP-complete minimum-conductance problem. In this section we show that Algorithms 3 and 4 can be used instead of APR to find low-conductance cuts.

Let w_{ij} be the weight of the edge between two neighbor nodes $i \sim j$. We define the conductance of a subset of nodes $S \subset \mathcal{V}$ as

$$\Phi(S) := \frac{\sum_{i \in S} \sum_{j \in \mathcal{V} \setminus S, j \sim i} w_{ij}}{\min\left(\text{vol}(S), \text{vol}(\mathcal{V} \setminus S)\right)}.$$

The minimum-conductance problem is defined as

$$\Phi(\mathcal{G}) := \min_{S \subset V} \Phi(S).$$

Let p_k be the output of Algorithm 1 with input value α and let r_k be the residual of (1) calculated by Algorithm 1. Moreover, we define $C_\alpha \subseteq C$ such that $\text{vol}(C_\alpha) \geq \text{vol}(C)/2$. According to Theorem 5 in [3], we can use the output of Algorithm 1 as an input to a sweep procedure (see Section 2.2 in [3]) to produce clusters of low-conductance. More precisely, the sweep procedure sorts the indices in $\text{supp}(p_k)$ in decreasing order w.r.t. to the values of the components of $D^{-1}p_k$. Let $i_1, i_2, \ldots, i_{|H_k|}$ be the sorted indices, where $H_k = \text{supp}(p_k)$. Using the sorted indices the sweep procedure generates a collection of sets $\mathcal{S}_j := \{i_1, i_2, \ldots, i_j\}$ for each $j \in \{1, 2, \ldots, |H_k|\}$. Provided that there exists a subset of nodes C that satisfies $\Phi(C) \leq \alpha/10$ and $\text{vol}(C) \leq 2\text{vol}(\mathcal{G})/3$, s is initialized within C_α and $\rho = 1/(10\text{vol}(C))$ then from Theorem 5 in [3] we get $\min_{j \in \{1,2,\ldots,|H_k|\}} \Phi(\mathcal{S}_j) \leq \sqrt{135 \log(m)\alpha}$. Theorem 5 in [3] suggests setting $\alpha = 10\Phi(\mathcal{G})$ which gives us $\min_{j \in \{1,2,\ldots,|H_k|\}} \Phi(\mathcal{S}_j) \leq \sqrt{1350\Phi(\mathcal{G})\log(m)}$. Similar guarantees can be proved if we replace Algorithm 1 with Algorithms 3 or 4.

Theorem 2. *Let $\rho = 1/(10\text{vol}(C))$, where C satisfies $\Phi(C) \leq \alpha/10$ and $\text{vol}(C) \leq 2\text{vol}(\mathcal{G})/3$, $\alpha = 10\Phi(\mathcal{G})$ and s is set within a subset of nodes $C_\alpha \subseteq C$. Algorithms 3 or 4 with $\rho' = \rho/2$ can be used instead of Algorithm 1 to find a set with conductance at most $\sqrt{1350\Phi(\mathcal{G})\log(m)}$.*

The proof is based on the fact that Algorithms 3 and 4 satisfy the invariance property of Algorithm 1 (see Section 3 in [3]). Moreover, all algorithms at termination satisfy $\|D^{-1/2}\nabla f(q_k)\|_\infty \leq \rho\alpha$. The proof is given in the appendix.

Page-Rank-Nibble One issue with the result in Theorem 2 is that there is no lower bound on the volume of the output cluster which is obtained from the sweep procedure. This implies that a very small set might be found. In Section 6 in [3] a procedure called Page-Rank-Nibble is proposed which deals with this problem. Page-Rank-Nibble makes a single call to Algorithm 1. We show that Algorithm 3 or 4 can be used instead. We describe the main result about Page-Rank-Nibble from Section 6 in [3], and we refer the reader to [3] for a more detailed discussion of the algorithm.

Let $\phi \in [0, 1]$ be a parameter and let us assume that there exists $C \subset \mathcal{V}$ such that $\text{vol}(C) \leq \text{vol}(\mathcal{G})/2$ and $\Phi(C) \leq \phi^2/(22500\log^2(100m))$. Page-Rank-Nibble makes a single call to Algorithm 1 and uses its output to produce the sweep sets, which were defined above. Theorem 7 in [3] suggests that if Algorithm 1 is initialized with $\alpha = \phi^2/(225\log(100m^{1/2}))$ and s is set in C_α then there exists some $b \in [1, \lceil \log m \rceil]$ such that if $\rho \leq (2^b 48\lceil \log m \rceil)^{-1}$ then at least one sweep set $\mathcal{S}_j$ satisfies $\Phi(\mathcal{S}_j) \leq \phi$, $2^{b-1} < \text{vol}(\mathcal{S}_j) < 2\text{vol}(\mathcal{G})/3$ and $\text{vol}(\mathcal{S}_j \cap C) > 2^{b-2}$. Similarly to Theorem 2, by replacing Algorithm 1 with Algorithm 3 or 4 we can obtain the same guarantees for Page-Rank-Nibble as in Theorem 7 in [3]. The proof is similar to that of Theorem 2 and we therefore omit it.

Theorem 3. *Let $\phi \in [0, 1]$, $\alpha = \phi^2/(225\log(100m^{1/2}))$ and s is set within a subset of nodes $C_\alpha \subseteq C$, where C satisfies $\text{vol}(C) \leq \text{vol}(\mathcal{G})/2$ and $\Phi(C) \leq \phi^2/(22500\log^2(100m))$. There exists $b \in [1, \lceil \log m \rceil]$ such that if $\rho \leq (2^b 48\lceil \log m \rceil)^{-1}$ then Algorithm 3 or 4 with $\rho' = \rho/2$ can be used in Page-Rank-Nibble to find a set S that satisfies $\Phi(S) \leq \phi$, $2^{b-1} < \text{vol}(S) < 2\text{vol}(\mathcal{G})/3$ and $\text{vol}(S \cap C) > 2^{b-2}$.*

9 Experiments

This section describes experimental results for our implementations of the algorithms described in this paper.

The experiments are performed on a single thread of a 64-core machine with four 2.4 GHz 16-core AMD Opteron 6278 processors. The implementations are written using C++ code and compiled with the g++ compiler version 4.8.0. We use a set of undirected, unweighted real-world graphs from the Stanford Network Analysis Project (`http://snap.stanford.edu/data`), whose sizes are shown in Table 1. All implementations will be part of the Ligra project (`https://github.com/jshun/ligra`). We present the performance of greedy and heuristic

Input Graph	Num. Vertices	Num. Edges[†]
wiki-Talk	2,394,385	4,659,565
soc-LJ	4,847,571	42,851,237
cit-Patents	6,009,555	16,518,947
com-Orkut	3,072,627	117,185,083

Table 1: Graph inputs. [†]Number of unique undirected edges.

versions of APR, ISTA and block-ISTA. In particular, in the following figures APR GREEDY is Algorithm 2 where in step 3 we select the i'th coordinate with the largest partial derivative $\nabla_i f(q_k)$ in absolute value. APR HEURISTIC is Algorithm 2 where we select approximately the i'th coordinate with the largest $\nabla_i f(q_k)$ in absolute value. In particular, a priority queue of coordinates is maintained which initially contains the starting vertex only. On each iteration we select the highest-priority coordinate in the queue and update the coordinate and its neighbors accordingly. For each neighbor, insert it in the queue if it is above the threshold with priority equal to the chosen coordinate. Note that this is a heuristic because we select coordinates based on their priority when they are initially inserted in the queue, and do not update their priorities later on. It is important to mention that the heuristic versions of the algorithms are guaranteed to converge in theory but not with linear convergence rate. However, there exist examples where one can maintain the linear convergence rate, as discussed in Section 5 in [10]. The Cheeger-like guarantees on the output quality hold for the heuristic versions as well. For block-ISTA we select 20% of S_k $\forall k$ in a greedy fashion. For single-ISTA we set $\tau_k = 1$ $\forall k$ and we also implement greedy and heuristic versions of it. For the greedy versions of block-ISTA we select coordinate based on the absolute values of $\nabla_i f(q_k) + \rho\alpha d_i^{1/2}$, as stated in Algorithm 4.

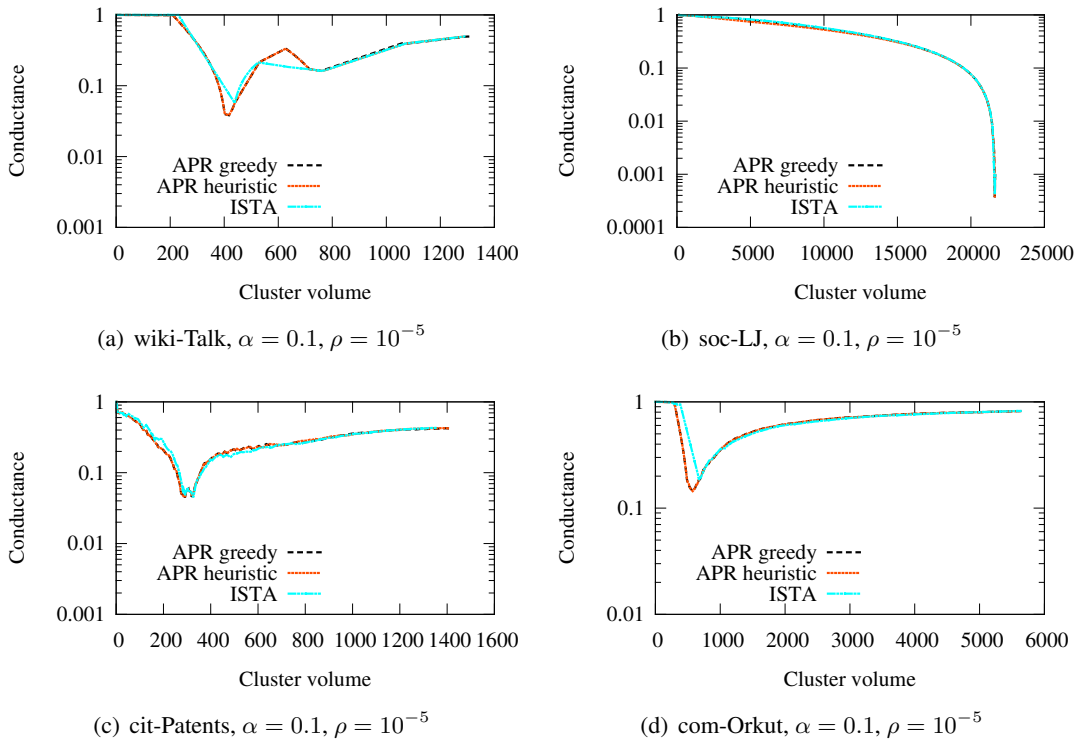
For all experiments we set $s = e_v$, where the coordinate/node v is chosen based on a search of over 10^4 starting nodes. We used the starting vertex that gave the best conductance. We conduct all experiments by fixing $\alpha = 0.1$ and choose the ρ values empirically such that we get clusters with at least 100 nodes each. This agrees with the observations in [17] regarding the size of local clusters in large-scale graphs.

9.1 ℓ_1-reg. PR $\approx$ APR

We demonstrate that ℓ_1-reg. PR problem achieves in practice similar graph cut guarantees as APR. We use the same sweep procedure as the one described in Section 2.2 in [3] for the original APR algorithm, which is based on the conductance criterion. In Figure 1 we present the conductance criterion (y-axis) versus the volume of the clusters (x-axis) produced by the sweep procedure in increasing order. All algorithms obtain approximately the same conductance value after the sweep procedure. The number of non-zeros of the output p_k for each algorithm is given in Table 2. Notice that the output of the ℓ_1-reg. PR problem, which is obtained by ISTA, has at most the same number of non-zeros as the greedy and the heuristic versions of APR.

Input Graph	APR GREEDY	APR HEUR.	ISTA
wiki-Talk	326	334	326
soc-LJ	159	159	159
cit-Patents	210	211	198
com-Orkut	447	448	442

Table 2: Number of non-zeros for the output solution p_k of each algorithm for the four experiments in Figure 1.



(a) wiki-Talk, $\alpha = 0.1$, $\rho = 10^{-5}$

(b) soc-LJ, $\alpha = 0.1$, $\rho = 10^{-5}$

(c) cit-Patents, $\alpha = 0.1$, $\rho = 10^{-5}$

(d) com-Orkut, $\alpha = 0.1$, $\rho = 10^{-5}$

Figure 1: Conductance vs. cluster volume. The axes of all plots are in log-scale. This figure shows the conductance criterion for the clusters which are produced by the sweep procedure applied on the output of each algorithm. The volume of the clusters is shown in increasing size.

9.2 Running time

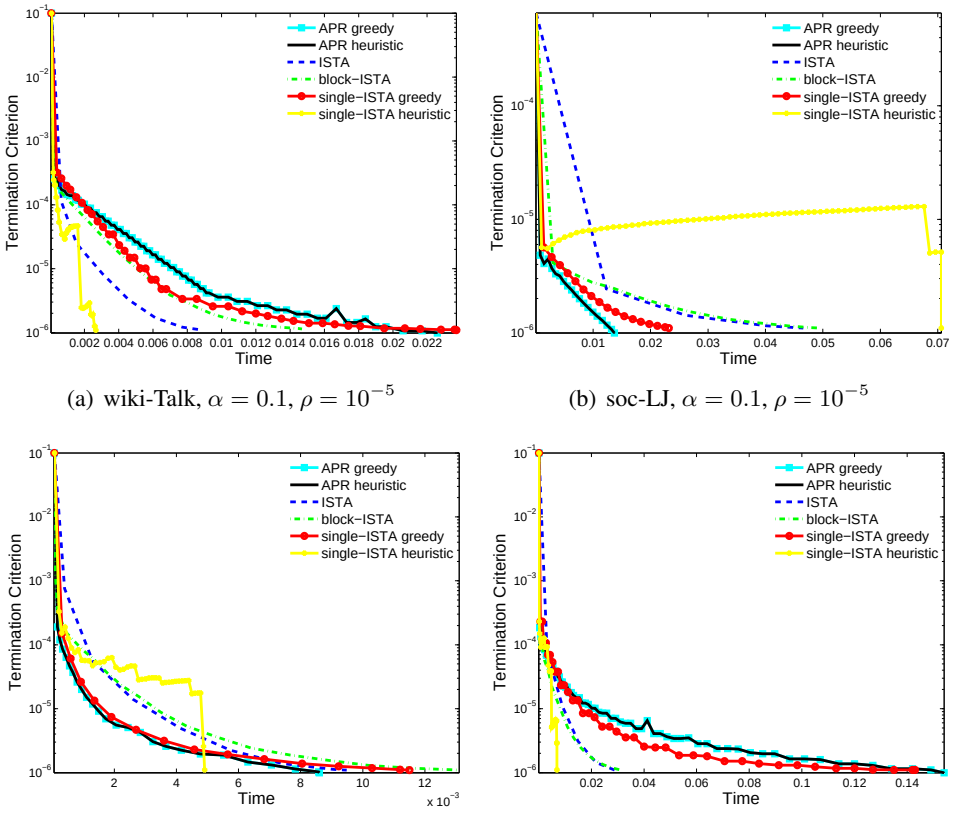
In Figure 2 we plot the value of the quantity $\|D^{-1/2}\nabla f(q_k)\|_\infty$ against the running time of each algorithm. The quantity $\|D^{-1/2}\nabla f(q_k)\|_\infty$ is used as a termination criterion for all algorithms. Notice that all ISTA-based algorithms are quite efficient and occasionally faster than APR. Overall, the fastest algorithms are the ones that are based on heuristics for choosing the next coordinates/nodes to be updated.

10 Conclusions

We have examined a popular local graph diffusion algorithm, and we have explained many of its properties using optimization tools. This bridges the gap between two research communities, that of graph clustering and of optimization, and we expect that the insights presented in this paper will help other researchers to understand and develop new algorithms for local graph clustering applications. For example, using this optimization perspective, we discuss two standard algorithms in optimization, i.e., an iterative soft-thresholding algorithm (ISTA) and a block-ISTA, which can be applied for graph clustering applications, while maintaining a running time proportional to the size of the output set. We presented empirical results demonstrating that optimization algorithms such as ISTA and block-ISTA for graph clustering can be very efficient in practice.

Acknowledgements

Funding from the DARPA XDATA and GRAPHS programs is gratefully acknowledged. J. Shun was supported by the Miller Institute for Basic Research in Science at UC Berkeley.



(a) wiki-Talk, $\alpha = 0.1$, $\rho = 10^{-5}$

(b) soc-LJ, $\alpha = 0.1$, $\rho = 10^{-5}$

(c) cit-Patents, $\alpha = 0.1$, $\rho = 10^{-5}$

(d) com-Orkut, $\alpha = 0.1$, $\rho = 10^{-5}$

Figure 2: Termination criterion, i.e., $\|D^{-1/2}\nabla f(q_k)\|_\infty$, vs. running time. The axes of all plots are in semi-log-vertical scale.

References

[1] Andersen, R. and Lang, K. An algorithm for improving graph partitions. *SODA '08 Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 651–660, 2008.

[2] Andersen, R. and Peres, Y. Finding sparse cuts locally using evolving sets. *STOC '09 Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 235–244, 2009.

[3] Andersen, R., Chung, F., and Lang, K. Local graph partitioning using pagerank vectors. *FOCS '06 Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 475–486, 2006.

[4] Argyriou, A., Foygel, R., and Srebro, N. Sparse prediction with the k-support norm. *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pp. 1–9, 2012.

[5] Arora, S., Rao, S., and Vazirani, U. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2:5), 2009.

[6] Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. Optimization with sparsity-inducing penalties. *Found. Trends Mach. Learn.*, 4(1):1–106, 2011.

[7] Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, pp. 183–202, 2009.

[8] Chung, F. A local graph partitioning algorithm using heat kernel pagerank. *Internet Mathematics*, 6(3): 315–330, 2009.

[9] Chung, F. and Simpson, O. Computing heat kernel pagerank and a local clustering algorithm. *25th International Workshop, IWOCA 2014*, pp. 110–121, 2014.

[10] Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Nearest neighbor based greedy coordinate descent. *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, 2011.

[11] Gleich, D. F. and Mahoney, M. W. Anti-differentiating approximation algorithms: A case study with min-cuts, spectral, and flow. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1018–1025, 2014.

[12] Grady, L. and Schwartz, E. L. Isoperimetric partitioning:a new algorithm for graph partitioning. *SIAM Journal on Scientific Computing*, 27(6):1844–1866, 2006.

[13] Hall, K. M. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.

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

[15] Kloster, K. and Gleich, D. F. Heat kernel based community detection. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1386–1395, 2014.

[16] Leighton, T. and Rao, S. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pp. 422–431, 1988.

[17] Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet of Mathematics*, 6(1):29–123, 2011.

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

[19] Nesterov, Y. *Introductory Lecture Notes On Convex Optimization. A Basic Course*. Kluver, Boston, 2004.

[20] Nutini, J., Schmidt, M., Laradji, I. H., Friedlander, M., and Koepke, H. Coordinate descent converges faster with the gauss-southwell rule than random selection. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1632–1641, 2015.

[21] Orecchia, L. and Zhu, Z. A. Flow-based algorithms for local graph clustering. *SODA '14 Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1267–1286, 2014.

[22] Parikh, Neal and Boyd, Stephen. Proximal algorithms. *Foundations and Trends in optimization*, 1(3): 123–231, 2013.

[23] Pothen, A., Simon, H. D., and Liou, K. P. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[24] Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program. Ser. A*, 144(1):1–38, 2014.

[25] Spielman, D. A. and Teng, S. H. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Scientific Computing*, 42(1):1–26, 2013.

[26] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[27] Wang, Y. X., Sharpnack, J., Smola, A., and Tibshirani, R. Trend filtering on graphs. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pp. 1042–1050, 2014.

A Proofs

Proof of Lemma 1. We will prove this statement by induction. Let us assume that at the kth iteration we have $q_k \geq 0$ and $\nabla f(q_k) \leq 0$. Further, let assume that there exists coordinate i such that $\nabla_i f(q_k) < -\rho\alpha d_i^{1/2}$, otherwise, the termination criterion is satisfied. Algorithm 2 chooses one coordinate which satisfies $\nabla_i f(q_k) < -\rho\alpha d_i^{1/2}$. Then from Step 4 of Algorithm 2 we have that $q_{k+1} \geq q_k$. Moreover, from Steps 5,6 and 7 we have that $\nabla_i f(q_k) < \nabla_i f(q_{k+1}) < 0$, $\nabla_j f(q_{k+1}) < \nabla_j f(q_k) \leq 0$ for each j such that $i \sim j$ and $\nabla_j f(q_{k+1}) = \nabla_j f(q_k) \leq 0$ for each j such that $i \nsim j$. Hence, $\nabla f(q_{k+1}) \leq 0$. Let $q_0 = 0$ and $s \geq 0$ in Algorithm 2. Then $\nabla f(q_0) = -\alpha s \leq 0$. We conclude that $q_{k+1} \geq q_k \geq 0$ and $\nabla f(q_k) \leq 0\ \forall k$. □

Proof of Theorem 1. Define

$$\tilde{f}(q; q_k) := f(q_k) + (q - q_k)^T \nabla f(q_k) + \frac{1}{2}\|q - q_k\|_2^2,$$
$$\tilde{\psi}(q; q_k) := \rho\alpha\|D^{1/2}q\|_1 + \tilde{f}(q; q_k).$$

It is easy to see that

$$\arg\min_q \tilde{\psi}(q; q_k) = \arg\min_q \rho\alpha\|D^{1/2}q\|_1 + \frac{1}{2}\|q - (q_k - \nabla f(q_k))\|_2^2,$$

and hence

$$q(i) = \mathbf{prox}_{\rho\alpha d_i^{1/2}\|.\|_1}\left(q_k(i) - \nabla_i f(q_k)\right),$$

where **prox** is the proximal operator [22]. Now let us define the sets

$$S_k := \{i \in [n] \mid q_k(i) - \nabla_i f(q_k) \geq \rho\alpha d_i^{1/2}\},$$
$$\widehat{S}_k := \{i \in [n] \mid -\rho\alpha d_i^{1/2} < q_k(i) - \nabla_i f(q_k) < \rho\alpha d_i^{1/2}\},$$
$$\widetilde{S}_k := \{i \in [n] \mid q_k(i) - \nabla_i f(q_k) \leq -\rho\alpha d_i^{1/2}\}.$$

For convenience we rewrite below ISTA from [7]. To show that Algorithm 3 and 5 are equivalent, it suffices to

Algorithm 5 ISTA for (4)

1: **Initialize**: $\rho > 0$, $q_0 = 0$, thus $\nabla f(q_0) = -\alpha D^{-1/2}s$
2: **while** termination criteria are not satisfied **do**
3: $q_{k+1}(i) = \mathbf{prox}_{\rho\alpha d_i^{1/2}\|.\|_1}\left(q_k(i) - \nabla_i f(q_k)\right),\ \forall i$, whose closed-form solution is given by

$$q_{k+1}(i) = \begin{cases} q_k(i) - (\nabla_i f(q_k) + \rho\alpha d^{1/2}(i)) & \text{if } i \in S_k \\ q_k(i) - (\nabla_i f(q_k) - \rho\alpha d^{1/2}(i)) & \text{if } i \in \widetilde{S}_k \\ 0 & \text{if } i \in \widehat{S}_k. \end{cases}$$

4: Calculate new gradient $\nabla f(q_{k+1})$.
5: $k = k + 1$
6: **end while**
7: **return** $p_k := D^{1/2}q_k$

show that $\widetilde{S}_k = \emptyset, \forall k$. We will prove the result by induction. Let us assume that at iteration k we have a $q_k \geq 0$, $\nabla f(q_k) \leq 0$ and $\nabla_i f(q_k) \leq -\rho\alpha d_i^{1/2}\ \forall i \in S_k$. As a result of the first two assumptions, we have $\widetilde{S}_k = \emptyset$ and $S_k \cap \widehat{S}_k = [n]$. Hence, Step 3 of ISTA Algorithm 5 can be simplified as

$$q_{k+1}(i) = \begin{cases} q_k(i) - (\nabla_i f(q_k) + \rho\alpha d^{1/2}(i)) & \text{if } i \in S_k \\ 0 & \text{if } i \in \widehat{S}_k \end{cases}. \tag{13}$$

Define $\Delta q_k := -I_{S_k}^T \left(\nabla f(q_k) + \rho\alpha D^{1/2} e\right)$, where I_{S_k} is defined in Section 3. Consequently, at iteration k, the new gradient components are updated as follows

$$\nabla_i f(q_{k+1}) = \begin{cases} -\rho\alpha d_i^{1/2} - \frac{1-\alpha}{2}\Big([I_{S_k}\Delta q_k]_i + \frac{1}{d_i^{1/2}} \sum\limits_{l\sim i, l\in S_k} \frac{A_{i,l}[I_{S_k}\Delta q_k]_l}{d_l^{1/2}}\Big), & i \in S_k \\ \nabla_i f(q_k) - \frac{1-\alpha}{2d_i^{1/2}} \sum\limits_{l\sim i, l\in S_k} \frac{A_{i,l}[I_{S_k}\Delta q_k]_l}{d_l^{1/2}}, & i \in \widehat{S}_k \text{ and } i \sim S_k \\ \nabla_i f(q_k), & i \in \widehat{S}_k \text{ and } i \nsim S_k \end{cases}, \quad (14)$$

where A is the adjacency matrix of the given graph. By induction hypothesis and noticing that $\Delta q_k \geq 0$ and $A_{i,l} \geq 0, \forall i, l$, it is easy to see that by (13), we have $q_{k+1} \geq 0$, and by (14), we get $\nabla f(q_{k+1}) \leq 0$. Hence, it follows that $\widetilde{S}_{k+1} = \emptyset$. In addition, for any $i \in S_k$, we get $\nabla_i f(q_{k+1}) \leq -\rho\alpha d_i^{1/2}$ and, as such, $i \in S_{k+1}$. In other words, once an index i enters the set S_k at iteration k, it will continue to stay in that set for all subsequent iterations, and so we always have $q_{k+1}(i) \geq q_k(i)$. In fact, the only indices entering S_{k+1} are those from $\widehat{S}_k$ which are neighbors of S_k (see (13) and (14) for $i \in \widehat{S}_k$) thus, $|S_k| \leq |S_{k+1}|$. In this case, suppose that $i \in \widehat{S}_k \cap S_{k+1}$. By (13), we have $q_{k+1}(i) = 0$, which combined with the definition of S_{k+1}, yields $\nabla_i f(q_{k+1}) \leq -\rho\alpha d_i^{1/2}$. As a result, we have $\nabla_i f(q_{k+1}) \leq -\rho\alpha d_i^{1/2}, \forall i \in S_{k+1}$. All is left to do is to start the iterations with the proper initial conditions, so that the base case of the induction holds. Set ρ small enough such that $\|s\|_\infty \geq \rho$. Now since $s \geq 0$, by choosing $q_0 = 0$, we have that $\nabla f(q_0) = -\alpha D^{1/2} s \leq 0$ and $\nabla_i f(q_0) \leq -\rho\alpha d_i^{1/2}\ \forall i \in S_0$. In addition, by such a choice of q_0, (13) as well as decreasing nature of $\widehat{S}_k$ imply that $q_{k+1} \geq q_k, \forall k$.

Finally, since $q_{k+1} \geq q_k\ \forall k$ then Algorithm 3 will not update more than $|\text{supp}(q_*)|$ coordinates. Thus, we have that $|S_k| \leq |\text{supp}(q_*)|\ \forall k$. □

Lemma 2. *Let $p := D^{1/2} q$, then the following is always true*

$$p + pr(\alpha, -\frac{1}{\alpha} D^{1/2} \nabla f(q)) = \alpha D^{1/2} Q D^{-1/2} s.$$

In other words, $p + pr(\alpha, -\frac{1}{\alpha} D^{1/2} \nabla f(q))$ is a constant.

Proof of Lemma 2. Recall that $f(q) = \frac{1}{2} q^T Q q - \alpha q^T D^{-1/2} s$, so $\nabla f(q) = Qq - \alpha D^{-1/2} s$, and $Q := D^{-1/2}\left\{D - \frac{1-\alpha}{2}(D + A)\right\} D^{-1/2}$. Let $pr(\alpha, s)$ be the solution to the Page-Rank system in (1), i.e., $pr(\alpha, s)$ satisfies the following

$$\frac{1-\alpha}{2}(I + AD^{-1}) pr(\alpha, s) + \alpha s = pr(\alpha, s). \quad (15)$$

After some manipulations, one can obtain

$$pr(\alpha, s) = \alpha D^{1/2} Q^{-1} D^{-1/2} s. \quad (16)$$

Thus

$$\begin{aligned} p + pr(\alpha, -\frac{1}{\alpha} D^{1/2} \nabla f(q)) &= p + \alpha D^{1/2} Q^{-1} D^{-1/2} (-\frac{1}{\alpha} D^{1/2} \nabla f(q)) \\ &= p - D^{1/2} Q^{-1} \nabla f(q) \\ &= p - D^{1/2} Q^{-1} (Qq - \alpha D^{-1/2} s) \\ &= \alpha D^{1/2} Q^{-1} D^{-1/2} s. \end{aligned}$$

□

Proof of Theorem 2. The proof of Theorem 5 in [3] is based on two properties of Algorithm 1, which we discuss below, and the specific initialization of Algorithm 1 that is given in the preamble of Theorem 2.

Property 1 This property is claimed in Section 3 in [3]. Let r be the residual of (1), i.e., $r := (I - (1 - \alpha)W)p - \alpha s$ and $pr(\alpha, s)$ be the solution to the Page-Rank system for some vector s. Let $p_k\ \forall k$ be generated by Algorithm 1 and let us define $\tilde{r}_k := -(1/\alpha) r_k$. Then Algorithm 1 satisfies the invariance property

$$p_{k+1} + pr(\alpha, \tilde{r}_{k+1}) = p_k + pr(\alpha, \tilde{r}_k) \quad \forall k. \quad (17)$$

Property 2 According to the termination criterion of Algorithm 1 in (3) the residual r_k at termination satisfies

$$\|D^{-1}r_k\|_\infty \leq \rho\alpha. \tag{18}$$

Let us explain where these two properties are used in Theorem 5 in [3]. To prove Theorem 5 in [3] the authors used the results of Theorem 4 and Theorem 2 in [3]. Theorem 4 uses **Property 1** and **Property 2**. Theorem 2 makes the following calling sequence Theorem 2 $\leftarrow$ Theorem 3 $\leftarrow$ Lemma 5 $\leftarrow$ Lemma 3, where the left arrow means "uses". The latter lemma uses **Property 1**. All other properties that are used to prove Theorem 5 are independent of Algorithm 1.

We will prove that Algorithms 3 and 4 satisfy **Property 1** and **Property 2**. Therefore, if they are initialized as mentioned in the preamble of Theorem 2 then they have the same graph cut guarantees as the original Algorithm 1.

We start by proving that Algorithms 3 and 4 satisfy the invariance property. Let $q_k \; \forall k$ obtained by Algorithm 3 or 4. Using $r_k = D^{1/2}\nabla f(q_k) \; \forall k$ and the definition $\tilde{r}_k = -(1/\alpha)r_k$ we get $-(1/\alpha)D^{1/2}\nabla f(q_k) = \tilde{r}_k$. Let us also define $p_k := D^{1/2}q_k \; \forall k$. Using Lemma 2 we get that Algorithms 3 and 4 satisfy

$$\begin{aligned} p_{k+1} + pr(\alpha, \tilde{r}_{k+1}) &= p_{k+1} + pr(\alpha, -(1/\alpha)D^{1/2}\nabla f(q_{k+1})) \\ &= \alpha D^{1/2}QD^{-1/2}s \\ &= p_k + pr(\alpha, -(1/\alpha)D^{1/2}\nabla f(q_k)) \\ &= p_k + pr(\alpha, \tilde{r}_k)). \end{aligned}$$

Thus, **Property 1** (17) is satisfied for Algorithms 3 and 4.

From the definition of the termination condition of Algorithms 3 and 4, we see that they return a q_k which satisfies $\|D^{-1/2}\nabla f(q_k)\|_\infty \leq (1+\epsilon)\rho\alpha$, where $\epsilon \in (0,1)$. Using $r_k = D^{1/2}\nabla f(q_k) \; \forall k$ we get that $\|D^{-1/2}\nabla f(q_k)\|_\infty \leq (1+\epsilon)\rho\alpha$ is equivalent to $\|D^{-1}r_k\|_\infty \leq (1+\epsilon)\rho\alpha$, which is the termination criterion (18) of Algorithm 1 up to a factor $1+\epsilon$. By initializing Algorithm 3 or Algorithm 4 with $\rho' = \rho/2$, we satisfy **Property 2** (18). Since Algorithms 3 and 4 satisfy both **Property 1** and **Property 2**, Theorem 5 in [3] holds for Algorithms 3 and 4 as well. □