SIAM J. MATH. DATA SCI.
Vol. 1, No. 2, pp. 333–355

Metric-Constrained Optimization for Graph Clustering Algorithms*

Nate Veldt†, David F. Gleich‡, Anthony Wirth§, and James Saunderson¶

Abstract. We outline a new approach for solving linear programming relaxations of NP-hard graph clustering problems that enforce triangle inequality constraints on output variables. Extensive previous research has shown that solutions to these relaxations can be used to obtain good approximation algorithms for clustering objectives. However, these are rarely solved in practice due to their high memory requirements. We first prove that the linear programming relaxation of the correlation clustering objective is equivalent to a special case of a well-known problem in machine learning called metric nearness. We then develop a general solver for metric-constrained linear and quadratic programs by generalizing and improving a simple projection algorithm, originally developed for metric nearness. We give several novel approximation guarantees for using our approach to find lower bounds for challenging graph clustering tasks such as sparsest cut, maximum modularity, and correlation clustering. We demonstrate the power of our framework by solving relaxations of these problems involving up to 10^7 variables and 10^{11} constraints.

Key words. graph clustering, correlation clustering, sparsest cut, modularity, metric learning, projection methods

AMS subject classifications. 05C50, 05C85, 65K05, 68W25, 90C35

DOI. 10.1137/18M1217152

1. Introduction.

Clustering is the task of identifying groups of closely related entities in a large dataset, and is one of the most fundamental problems in data analysis. For problems in which the dataset is specifically modeled by a graph, this problem is referred to as graph clustering or community detection. In this paper we consider a special class of graph clustering algorithms that come with theoretically rigorous approximation guarantees but rely on solving expensive linear programs involving $\Theta(n^3)$ *metric constraints* of the form $x_{ij} \leq x_{ik} + x_{jk}$, where x_{ij} represents some form of distance between two nodes i and j in a graph of n nodes.

One of the best examples of a metric-constrained linear program is the canonical relaxation of the correlation clustering integer linear program [6, 13]. Correlation clustering is a framework for partitioning datasets characterized by pairwise similarity and dissimilarity scores,

*Received by the editors September 28, 2018; accepted for publication (in revised form) March 26, 2019; published electronically June 4, 2019.

http://www.siam.org/journals/simods/1-2/M121715.html

Funding: The first author was supported by NSF award IIS-1546488, and the second author by NSF awards CCF-1149756, IIS-1422918, IIS-1546488, NSF Center for Science of Information STC, CCF-0939370, DOE DE-SC0014543, NASA, and the Sloan Foundation. The third author was supported by the Melbourne School of Engineering.

†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (lveldt@purdue.edu).

‡Department of Computer Science, Purdue University, West Lafayette, IN 47907 (dgleich@purdue.edu).

§School of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3052, Australia (awirth@unimelb.edu.au).

¶Department of Electrical and Computer Systems Engineering, Monash University, Clayton, VIC 3800, Australia (james.saunderson@monash.edu).

and can be equivalently viewed as a clustering problem on signed graphs. The framework has been applied to problems in bioinformatics [27], social network analysis [48], image segmentation [31], and many other domains. Correlation clustering has been extensively studied from a mathematical and computational perspective, and many of its best-known approximations rely on rounding a metric-constrained linear programming (LP) relaxation [2, 14, 41, 42, 48]. Unfortunately, these algorithms are rarely implemented due to the high memory requirement of traditional solvers. Other graph clustering objectives with well-studied metric-constrained relaxations include sparsest cut [35], cluster deletion [48], modularity clustering [1], and maximum cut [5, 40].

In our work we develop a new approach for solving metric-constrained linear programs based on projection methods, which come with a significantly smaller memory footprint than standard solvers. This approach allows data scientists to implement clustering approximation algorithms that come with strong guarantees, but were previously only viewed as theoretical results. The starting point in our research is the observation that the LP relaxation for correlation clustering is equivalent to a special case of the ℓ_1 metric nearness problem [12]. Based on this, we develop a general strategy for metric-constrained optimization that is related to the iterative triangle-fixing algorithms that Dhillon, Sra, and Tropp developed for metric nearness [19]. This approach considers a regularized linear program that is closely related to the original LP, but can be solved using Dykstra's projection method [21]. We generalize and improve the techniques of Dhillon, Sra, and Tropp to apply them more broadly to any metric-constrained linear or quadratic program. Our method comes with a more robust stopping criterion and an entrywise rounding procedure when the solver is close to convergence, which improves the runtime and effectiveness of the solver. We apply this framework to solving metric-constrained LP relaxations of correlation clustering and sparsest cut, and prove several novel guarantees regarding lower bounds to these objectives.

In practice we are able to solve the Leighton–Rao relaxation for sparsest cut on graphs with up to thousands of nodes. For correlation clustering, we solve dense problems involving up to 11 thousand nodes, 6×10^7 variables, and 7×10^{11} constraints. We additionally show how our method can be used to obtain good approximation guarantees and improved algorithmic results for maximum modularity clustering.

2. Background and related work. Many theoretical approximation algorithms for clustering rely on solving metric-constrained LPs [2, 35, 41, 48], but limited work has addressed practical implementations. Wirth noted that the LP relaxation of correlation clustering can be solved more efficiently by using a multicommodity flow formulation of the problem, though this is still very expensive in practice [51]. Van Gael and Zhu employed an *LP chunking* technique which allowed them to solve this relaxation on graphs with nearly 500 nodes [46]. The LP rounding algorithm of Charikar, Guruswami, and Wirth [13] inspired others to use metric-constrained LPs for modularity clustering [1] and joint-clustering of image segmentations [24, 49]. In practice these algorithms scale to only a few hundred nodes when a full set of $O(n^3)$ constraints is included. In the case of sparsest cut, Lang and Rao developed a practical algorithm closely related to the original Leighton–Rao algorithm [32], which was later evaluated empirically by Lang, Mahoney, and Orecchia [33]. However, the algorithm only heuristically solves the underlying multicommodity flow problem, and therefore does not

satisfy the same theoretical guarantees.

Recent work has shown that correlation clustering problems can be solved to *optimality* more efficiently using MaxSAT-based integer programming solvers [9, 38]. While these typically do not scale past a few hundred nodes, Miyauchi, Sonobe, and Sukegawa were able to solve *sparse* correlation clustering problems involving up to 2500 nodes. Their approach relied crucially on the observation that for sparse problems, a significant number of the metric constraints do not need to be explicitly included by the integer program [38], though this approach does not apply to dense instances.

There also exist approaches which provide lower bounds and approximate solutions for correlation clustering by considering the dual of LP relaxations that are not the canonical metric-constrained LP [34, 44]. These approaches can be combined with fast local clustering heuristics to obtain good output clusterings, but cannot be used to implement correlation clustering approximation algorithms that come with the best-known a priori guarantees. To realize the power of these approximation guarantees, we need scalable solvers for the canonical LP.

3. Graph clustering and metric-constrained linear programs. Formally, we define a metric-constrained optimization problem to be an optimization problem involving constraints of the form $x_{ij} \leq x_{ik} + x_{jk}$, where x_{ij} is a nonnegative variable representing a distance score between two objects i and j in a given dataset. Optimization problems of this form arise very naturally in the study of graph clustering objectives, since any strict clustering $\mathcal{C}$ for a graph $G = (V, E)$ is in one-to-one correspondence with a set of binary variables x_{ij} satisfying triangle inequality constraints:

$$\begin{cases} x_{ij} \in \{0,1\} & \text{for all } i,j \text{ and} \\ x_{ij} \leq x_{ik} + x_{jk} & \text{for all } i,j,k \end{cases} \iff \exists\ \mathcal{C} \text{ such that } x_{ij} = \begin{cases} 0 & \text{if } i,j \text{ are together in } \mathcal{C}, \\ 1 & \text{otherwise}. \end{cases}$$

Metric-constrained *linear* programming relaxations have been introduced and studied for a large number of graph clustering objectives, including modularity [1], cluster deletion [48], and maximum cut [40]. In our work we focus, in particular, on metric-constrained LP relaxations of correlation clustering and sparsest cut, as special case studies. We will show in this section that the correlation clustering relaxation is, in fact, equivalent to the so-called metric nearness problem (Theorem 3.1), which will be the starting point for our development of practical techniques for solving metric-constrained graph clustering relaxations.

3.1. Correlation clustering. Correlation clustering is an NP-hard problem for partitioning a signed graph $G = (V, W^+, W^-)$ [6, 52]. Each pair of distinct nodes i and j in G possesses two nonnegative weights, $w_{ij}^+ \in W^+$ and $w_{ij}^- \in W^-$, indicating measures of similarity and dissimilarity, respectively. The goal is to cluster the nodes in a way that minimizes the total quantity of mistakes, where the mistake at pair (i,j) is w_{ij}^+ if i and j are separated, but w_{ij}^- if they are clustered together. The objective can be written formally as an integer linear program (ILP):

$$\begin{array}{lll} \text{minimize} & \sum_{i<j} w_{ij}^+ x_{ij} + w_{ij}^-(1 - x_{ij}) & \\ \text{subject to} & x_{ij} \leq x_{ik} + x_{jk} & \text{for all } i,j,k, \\ & x_{ij} \in \{0,1\} & \text{for all } i,j. \end{array} \tag{3.1}$$

An equivalent problem is to maximize the weight of agreements, which is the same at optimality, but different in terms of approximation algorithms [6]. When we relax the above ILP by replacing $x_{ij} \in \{0,1\}$ with the constraint $x_{ij} \in [0,1]$, this becomes a metric-constrained linear program that has been extensively studied in the literature. Semidefinite programming (SDP) relaxations for correlation clustering have also been studied [13, 43], and many heuristic algorithms have also been developed. However, the best approximation results for minimizing disagreements in both general weighted graphs (an $O(\log n)$ approximation [13, 17, 22]), and complete unweighted graphs (an approximation slightly better than 2.06 [14]) depend on first solving the LP relaxation. The best results for special weighted cases and deterministic pivoting algorithms also rely on solving the LP relaxation [48, 47, 41]. It is worth noting that the majority of these instances constitute *dense* correlation clustering problems.

3.2. Sparsest cut. A set $S \subset V$ of vertices in an n-node graph $G = (V, E)$ defines a cut by considering the set of edges with one endpoint in S and the other endpoint in the complement, $\bar{S} = V \setminus S$, of S. The sparsity of the cut defined by S is

$$\phi(S) = \frac{\mathbf{cut}(S)}{|S|} + \frac{\mathbf{cut}(S)}{|\bar{S}|} = \frac{n\,\mathbf{cut}(S)}{|S||\bar{S}|},$$

where $\mathbf{cut}(S)$ indicates the number of edges between S and $\bar{S}$. Leighton and Rao gave an $O(\log n)$-approximation for finding the minimum cut sparsity, $\phi^* = \min_{S \subset V} \phi(S)$ [35]. Their approach is equivalent to solving the following metric-constrained LP relaxation for ϕ^*:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} x_{ij} & \\ \text{subject to} & \sum_{i<j} x_{ij} = n & \\ & x_{ij} \le x_{ik} + x_{jk} & \text{for all } i,j,k, \\ & x_{ij} \ge 0 & \text{for all } i,j \end{array} \tag{3.2}$$

and rounding the solution into a cut. The Leighton–Rao $O(\log n)$ approximation for sparsest cut was for many years the best approximation for this problem, until Arora, Rao, and Vazirani developed an $O(\sqrt{\log(n)})$ approximation based on an SDP relaxation [4].

3.3. Metric nearness. The metric nearness problem [12] seeks the nearest *metric* matrix $\mathbf{X}^* = (x^*_{ij})$ to a *dissimilarity* matrix $\mathbf{D} = (d_{ij})$. Here, a dissimilarity matrix is a nonnegative, symmetric, zero-diagonal matrix, and a *metric* matrix is a dissimilarity matrix whose entries satisfy metric constraints. If M_n represents the set of metric matrices of size $n \times n$, then the ℓ_p version of the problem can be formalized as follows:

$$\mathbf{X}^* = \operatorname{argmin}_{\mathbf{X}\in M_n} \left(\sum_{i\neq j} w_{ij} \left|(x_{ij} - d_{ij})\right|^p \right)^{1/p}, \tag{3.3}$$

where $w_{ij} \ge 0$ is a weight indicating how strongly we wish $\mathbf{X}^*$ and $\mathbf{D}$ to coincide at entry ij. When $p = 1$, the problem can be cast as a linear program by introducing variables $\mathbf{M} = (m_{ij})$:

$$\begin{array}{lll} \text{minimize} & \sum_{i<j} w_{ij} m_{ij} & \\ \text{subject to} & x_{ij} \le x_{ik} + x_{jk} & \text{for all } i,j,k, \\ & x_{ij} - d_{ij} \le m_{ij} & \text{for all } i,j, \\ & d_{ij} - x_{ij} \le m_{ij} & \text{for all } i,j, \end{array} \tag{3.4}$$

where the last two constraints ensure that at optimality, $m_{ij} = |x_{ij} - d_{ij}|$. Our first theorem shows that LP (3.1) and LP (3.4) are, in fact, equivalent.

Theorem 3.1. *Consider an instance of correlation clustering $G = (V, W^+, W^-)$ and set $w_{ij} = |w_{ij}^+ - w_{ij}^-|$. Define an $n \times n$ matrix $\mathbf{D} = (d_{ij})$, where $d_{ij} = 1$ if $w_{ij}^- > w_{ij}^+$ and $d_{ij} = 0$ otherwise. Then $\mathbf{X} = (x_{ij})$ is an optimal solution to the LP relaxation of (3.1) if and only if $(\mathbf{X}, \mathbf{M})$ is an optimal solution for (3.4), where $\mathbf{M} = (m_{ij}) = (|x_{ij} - d_{ij}|)$.*

Proof. We will assume that at most one of w_{ij}^+, w_{ij}^- is positive, so every pair of nodes is either labeled similar or dissimilar. If this were not the case, we could introduce new edge weights $(\tilde{w}_{ij}^+, \tilde{w}_{ij}^-)$ for each pair ij defined to be $(\tilde{w}_{ij}^+, \tilde{w}_{ij}^-) = (w_{ij}^+ - w_{ij}^-, 0)$ when $w_{ij}^+ \geq w_{ij}^-$ and $(\tilde{w}_{ij}^+, \tilde{w}_{ij}^-) = (0, w_{ij}^- - w_{ij}^+)$ otherwise. This change of variables would alter the LP objective by only an additive constant, so the optimal LP solution would remain the same.

We equivalently consider an unsigned graph $G' = (V, E)$ with the same node set V and an adjacency matrix $\mathbf{A} = (A_{ij})$, where $A_{ij} = 1$ if $w_{ij}^- = 0$ and $A_{ij} = 0$ otherwise. If $w_{ij} = \max\{w_{ij}^+, w_{ij}^-\}$, the correlation clustering LP objective function can then be written as

$$\sum_{i<j} w_{ij} \left(A_{ij} x_{ij} + (1 - A_{ij})(1 - x_{ij})\right). \tag{3.5}$$

Define a dissimilarity matrix $\mathbf{D} = (d_{ij})$ by setting $d_{ij} = 1 - A_{ij}$. Notice that because $d_{ij} \in \{0, 1\}$ and $x_{ij} \in [0, 1]$, the key factor in the objective can be simplified in the following way:

$$(1 - d_{ij})x_{ij} + d_{ij}(1 - x_{ij}) = |x_{ij} - d_{ij}|,$$

and the LP relaxation of correlation clustering shown in (3.5) is equivalent to

$$\begin{array}{lll} \text{minimize} & \sum_{i<j} w_{ij}|x_{ij} - d_{ij}| & \\ \text{subject to} & x_{ij} \leq x_{ik} + x_{jk} & \text{for all } i, j, k, \\ & 0 \leq x_{ij} \leq 1 & \text{for all } i, j. \end{array} \tag{3.6}$$

The only difference between this objective and ℓ_1 metric nearness is that it includes the constraint $0 \leq x_{ij} \leq 1$. One can show that dropping these bounds does not change the optimal solution. The full proof follows from a lengthy case-by-case analysis. The idea, in short, is to consider any feasible solution that includes at least one variable $x_{ij} \notin [0, 1]$. We construct a new solution by replacing any $x_{ij} < 0$ with a new variable $x'_{ij} = 0$, and similarly replace $x_{ij} > 1$ with $x'_{ij} = 1$. By checking different cases, one can show that the resulting solution will still be feasible and have a strictly smaller objective score. ∎

4. Projection methods for metric-constrained optimization. We can state the linear programs in the previous section abstractly in the form

$$\min_{\mathbf{x}} \ \mathbf{c}^T \mathbf{x} \quad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{4.1}$$

where $\mathbf{A}$ has $\Theta(n^3)$ rows and $\Theta(n^2)$ columns, but is very sparse. Standard optimization software will be unable to solve these LPs for large values of n, due to memory constraints, so

we turn our attention to a simple projection method for solving a related quadratic program (QP), a regularization of (4.1):

$$\min_{\mathbf{x}} \; Q(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2\gamma}\mathbf{x}^T\mathbf{W}\mathbf{x} \quad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}. \tag{4.2}$$

In (4.2), above, $\mathbf{W}$ is a diagonal matrix with positive diagonal entries and $\gamma > 0$. When $\mathbf{W}$ is the identity matrix, there exists some $\gamma_0 > 0$ such that for all $\gamma > \gamma_0$, the optimal solution to the quadratic program corresponds to the minimum 2-norm solution of the original LP [37]. The dual of (4.2) is another quadratic program:

$$\max_{\mathbf{y}} \; D(\mathbf{y}) = -\mathbf{b}^T\mathbf{y} - \frac{\gamma}{2}(\mathbf{A}^T\mathbf{y} + \mathbf{c})^T\mathbf{W}^{-1}(\mathbf{A}^T\mathbf{y} + \mathbf{c}) \quad \text{subject to } \mathbf{y} \geq 0. \tag{4.3}$$

The core of our algorithm for solving (4.2) is Dykstra's projection method [21], which is typically presented as a way to solve the best approximation problem (BAP). Formally, let $C_i \subset \mathbb{R}^N$ for $i = 1, 2, \ldots, M$ be convex sets, and let $C = \cap_{i=1}^{M} C_i$ be their intersection (also convex). Given $\mathbf{z} \in \mathbb{R}^N$, the best approximation problem seeks $P_C(\mathbf{z})$, the *projection of* $\mathbf{z}$ *onto* C:

$$\mathbf{x}^* = P_C(\mathbf{z}) = \arg\min_{\mathbf{x} \in C} ||\mathbf{x} - \mathbf{z}||^2 \tag{4.4}$$

for some norm $||\cdot||$. Dykstra's method starts with an initial point $\mathbf{x}_0 = \mathbf{z}$ and iteratively performs simpler projections onto convex sets C_i in a way that is guaranteed to solve (4.4).

To cast (4.2) as an instance of BAP (4.4) so that we can apply Dykstra's method, we use a weighted norm defined by $\|\mathbf{v}\|_w^2 = (1/\gamma)\mathbf{v}^T\mathbf{W}\mathbf{v}$ and a starting vector $\mathbf{z} = -\gamma\mathbf{W}^{-1}\mathbf{c}$ so that

$$\|\mathbf{x} - \mathbf{z}\|_w^2 = \left\|\mathbf{x} + \gamma\mathbf{W}^{-1}\mathbf{c}\right\|_w^2 = \frac{1}{\gamma}\mathbf{x}^T\mathbf{W}\mathbf{x} + 2\mathbf{x}^T\mathbf{c} + \gamma\mathbf{c}^T\mathbf{W}^{-1}\mathbf{c}.$$

Since the final term is a constant, we see that the minimizers of (4.2) and (4.4) are the same if we are projecting onto half space constraints of the form $C_i = \{\mathbf{x} \in \mathbb{R}^N : \mathbf{a}_i^T\mathbf{x} \leq b_i\}$, where $\mathbf{a}_i$ is the ith row of the constraint matrix $\mathbf{A}$ in (4.2) and b_i is the ith entry of $\mathbf{b}$.

Algorithm 4.1 shows Dykstra's method applied to quadratic program (4.2). The method iteratively updates a set of primal and dual variables $\mathbf{x}$ and $\mathbf{y}$ that are guaranteed to converge to the optimal solutions of (4.2) and (4.3), respectively. When applied to strongly convex quadratic programs such as (4.2), Dykstra's method is equivalent to Hildreth's projection method [26], and is guaranteed to have a linear convergence rate [23]. Our full algorithmic approach takes Dykstra's method and includes a number of key features that allow us to efficiently obtain high-quality solutions to metric-constrained problems in practice. The first feature, a procedure for locally performing projections at metric constraints, is the key insight which led Dhillon, Sra, and Tropp to develop efficient algorithms for metric nearness [19]. In addition, we detail a sparse storage scheme for dual vectors, and include a more robust convergence check that leads to better constraint satisfaction and stronger optimality guarantees.

4.1. Efficient local updates. Projections of the form $\mathbf{x} := \mathbf{x} + \alpha\mathbf{W}^{-1}\mathbf{a}_i$ for $\mathbf{W}$ diagonal and α constant will change $\mathbf{x}$ by at most the number of nonzero entries of $\mathbf{a}_i$, the ith row of constraint matrix $\mathbf{A}$. In the case of triangle-inequality constraints, which dominate our constraint set, there are exactly three nonzero entries per constraint, so we perform each projection in a constant number of operations.

Algorithm 4.1 Dykstra's Method for Quadratic Programming.

Input: $\mathbf{A} \in \mathbb{R}^{N\times M}, \mathbf{b} \in \mathbb{R}^M, \mathbf{c} \in \mathbb{R}^N, \gamma > 0, \mathbf{W} \in \mathbb{R}^{N\times N}$(diagonal, positive definite)
Output: $\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}\in\mathcal{A}} Q(\mathbf{x})$, where $\mathcal{A} = \{\mathbf{x} \in \mathbb{R}^N : \mathbf{A}\mathbf{x} \le \mathbf{b}\}$
$\mathbf{y} := \mathbf{0} \in \mathbb{R}^M$, $\mathbf{x} := -\gamma\mathbf{W}^{-1}\mathbf{c}$, $k := 0$
while *not converged* **do**
$k := k + 1$
(Visit constraints cyclically): $i := (k-1) \bmod M + 1$
(Perform correction step): $\mathbf{x} := \mathbf{x} + y_i(\gamma\mathbf{W}^{-1}\mathbf{a}_i)$, where $\mathbf{a}_i$ is the ith row of $\mathbf{A}$
(Perform projection step): $\mathbf{x} := \mathbf{x} - \theta_i^+(\gamma\mathbf{W}^{-1}\mathbf{a}_i)$, where $\theta_i^+ = \frac{\max\{\mathbf{a}_i^T\mathbf{x}-b_i,0\}}{\gamma\mathbf{a}_i^T\mathbf{W}^{-1}\mathbf{a}_i}$
(Update dual variables): $y_i := \theta_i^+ \ge 0$

4.2. Sparse storage of dual variables. For constraint sets that include triangle inequalities for every triplet of nodes (i, j, k), the dual vector $\mathbf{y}$ will be of length $\Theta(n^3)$. Observe that the correction step in Algorithm 4.1 at constraint t will be nontrivial if and only if there was a nontrivial projection the last time the constraint was visited. In other words, θ_t^+ became nonzero in the update step of the previous round, and therefore $y_t > 0$. Note that each triplet (i, j, k) corresponds to three different metric constraints: $x_{ij} - x_{ik} - x_{jk} \le 0$, $x_{jk} - x_{ik} - x_{ij} \le 0$, and $x_{ik} - x_{ij} - x_{jk} \le 0$, and in each round at most one of these constraints will be violated, indicating that at least two dual variables will be zero. Dhillon, Sra, and Tropp concluded that $\binom{n}{3}$ floating point numbers must be stored in implementing Dykstra's algorithm for the metric nearness problem [18]. We further observe, especially for the correlation clustering LP, that, in practice, often for a large percentage of triplets (i, j, k), none of the three metric constraints are violated. Thus, we can typically avoid the worst-case $\Theta(n^3)$ memory requirement by storing $\mathbf{y}$ sparsely. In practice, therefore, we only store pairs (t, y_t) for $y_t > 0$, in either a dictionary with random access, or an array which the method traverses in the same cyclic order each round.

4.3. Robust stopping criteria. Many implementations of Dykstra's method stop when the change in vector $\mathbf{x}$ drops below a certain tolerance after one or more passes through the entire constraint set. However, Birgin and Raydan [10] noted that in some cases this may occur even when the iterates are far from convergence. Because we are applying Dykstra's method specifically to quadratic programming, we can obtain a much more robust stopping criterion by monitoring the dual objective function in a manner similar to the approach of Dax [16].

Let $(\mathbf{x}_k, \mathbf{y}_k)$ denote the pair of primal and dual vectors computed by Dykstra's method after k projections. We know that these vectors will converge to an optimal pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ such that $D(\hat{\mathbf{y}}) = \hat{Q} = Q(\hat{\mathbf{x}})$, where $\hat{Q}$ is the optimal objective for both the primal (4.2) and dual (4.3) quadratic programs. The Karush–Kuhn–Tucker (KKT) optimality conditions state that the pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is optimal for the primal (4.2) and dual (4.3) quadratic programs if and only if

1. $\mathbf{A}\hat{\mathbf{x}} \le \mathbf{b}$, **2.** $\hat{\mathbf{y}}^T(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}) = 0$, **3.** $(\mathbf{W}/\gamma)\hat{\mathbf{x}} = -\mathbf{A}^T\hat{\mathbf{y}} - \mathbf{c}$, **4.** $\hat{\mathbf{y}} \ge 0$.

The update step $y_i := \theta_i^+$ in Algorithm 4.1 guarantees that the last two KKT conditions are

always satisfied. In other words, at iteration k, variables $(\mathbf{x}_k, \mathbf{y}_k)$ can be shown to satisfy $\mathbf{y}_k \geq 0$ and $(\mathbf{W}/\gamma)\mathbf{x}_k = -\mathbf{A}^T\mathbf{y}_k - \mathbf{c}$. This means that $\mathbf{y}_k$ is always feasible for the dual (4.3). By weak duality we have the following lower bound on the optimal value of (4.2):

$$D(\mathbf{y}_k) = -\mathbf{b}^T\mathbf{y}_k - \frac{\gamma}{2}(\mathbf{A}^T\mathbf{y}_k + \mathbf{c})^T\mathbf{W}^{-1}(\mathbf{A}^T\mathbf{y}_k + \mathbf{c}) = -\mathbf{b}^T\mathbf{y}_k - \frac{1}{2\gamma}\mathbf{x}_k^T\mathbf{W}\mathbf{x}_k. \tag{4.5}$$

Dykstra's method is known to be equivalent to applying a coordinate-ascent procedure to a dual objective [45]. Thus, updating the dual variables via Dykstra's method provides a strictly increasing lower bound $D(\mathbf{y}_k)$ which converges to $\hat{Q}$ (see the work of Dax for more details [16]). Meanwhile, $Q(\mathbf{x}_k)$ does not necessarily upper bound $\hat{Q}$ since $\mathbf{x}_k$ is not necessarily feasible. However, $\mathbf{x}_k$ converges to the optimal primal solution, so as the algorithm progresses, the maximum constraint violation of $\mathbf{x}_k$ decreases to zero. In practice, once $\mathbf{x}_k$ has satisfied constraints to within a small enough tolerance, we treat $Q(\mathbf{x}_k)$ as an upper bound. After each pass through the constraints we check the primal-dual gap and maximum constraint violation, given by $\omega_k = [D(\mathbf{y}_k) - Q(\mathbf{x}_k)]/D(\mathbf{y}_k)$ and $\rho_k = \max_i(b_i - \mathbf{a}_i^T\mathbf{x}_k)$, respectively, and stop when both have fallen below a user-defined tolerance. Computing ρ_k exactly requires that we check $\Theta(n^3)$ constraints. However, this will be significantly faster than performing another pass through the constraints using Dykstra's method, since we are only checking constraint violations and not performing projection operations. In practice, we can also avoid increasing the running time too much by only performing a full constraint check every C iterations for some integer C, or simply returning *false* in the convergence check as soon as a violated constraint is encountered. Since this step only involves reading entries from a matrix, it is also easy to parallelize.

4.4. Entrywise rounding procedure. Our convergence check also includes a novel round-and-check procedure on the Dykstra iterate $\mathbf{x}_k$. Because $\mathbf{x}_k \to \hat{\mathbf{x}}$, we know that after a certain point, the maximum entrywise difference between $\hat{\mathbf{x}}$ and $\mathbf{x}_k$ (i.e., $\|\hat{\mathbf{x}} - \mathbf{x}_k\|_\infty$) will be small. Once ρ_k has dropped below a given tolerance and we know we are close to convergence, we can round every entry of $\mathbf{x}_k$ to r significant figures: $\mathbf{x}_r = round(\mathbf{x}_k, r)$. If $\mathbf{x}_k$ is close enough to optimality and we have chosen a good r, $\mathbf{x}_r$ will satisfy constraints to within the desired tolerance. We will then have an objective exactly or nearly equal to the best lower bound we have for $\hat{Q}$: $[D(\mathbf{y}_k) - Q(\mathbf{x}_r)]/D(\mathbf{y}_k) \leq \epsilon$. If $\mathbf{x}_r$ does not satisfy constraints or has a poor objective score, we simply continue with the original Dykstra iterate $\mathbf{x}_k$. The best choice of r and the benefit of this procedure will depend heavily on the user-defined tolerance, the specific objective, and the problem instance. In our experiments section we provide further details for how to set r and how this procedure performs for different objectives.

5. Approximation guarantees for clustering objectives. The results of Mangasarian [37] confirm that for all γ greater than some $\gamma_0 > 0$, the original linear program (4.1) and the quadratic regularization (4.2) have the same optimal solution. However, it is challenging to compute γ_0 in practice, and if we set γ to be too high, then this may lead to very slow convergence for solving QP (4.2) using projection methods. Dhillon, Sra, and Tropp [19] suggest ways to set γ for variants of the metric nearness problem based on empirical observations, but no approximation guarantees are provided. A key contribution in our work is a set of results, outlined in this section, which show how to set $\mathbf{W}$ and γ in order to obtain specific guarantees

for approximating the correlation clustering and sparsest cut objectives. These results hold for all $\gamma > 0$, whether larger or smaller than the unknown value γ_0.

5.1. Correlation clustering. Consider a correlation clustering problem on n nodes where each pair of nodes (i, j) is either strictly similar or strictly dissimilar, with a nonzero weight $w_{ij} > 0$. That is, exactly one of the weights (w_{ij}^-, w_{ij}^+) is positive and the other is zero. We focus on the LP relaxation for this problem given in the form of the ℓ_1 metric nearness LP (3.4). We slightly alter this formulation by performing a change of variables $y_{ij} = x_{ij} - d_{ij}$. The LP can then be written equivalently as

$$\begin{array}{lll} \text{minimize} & \sum_{i<j} w_{ij} m_{ij} & \\ \text{subject to} & y_{ij} - y_{ik} - y_{jk} \le b_{ijk} & \text{for all } i,j,k, \\ & y_{ij} \le m_{ij} & \text{for all } i,j, \\ & -y_{ij} \le m_{ij} & \text{for all } i,j, \end{array} \tag{5.1}$$

where $b_{ijk} = -d_{ij} + d_{ik} + d_{jk}$ is defined so that the implicit variables $x_{ij} = y_{ij} + d_{ij}$ satisfy triangle inequalities. To write this LP in the format of (4.1), we let $\mathbf{x} = \begin{bmatrix}\mathbf{y} & \mathbf{m}\end{bmatrix}^T$ and let $\mathbf{c} = \begin{bmatrix}\mathbf{0} & \mathbf{w}\end{bmatrix}^T$, where $\mathbf{y}, \mathbf{m}$ represent linearizations of the doubly-indexed (y_{ij}) and (m_{ij}) variables. The vector $\mathbf{w} = (w_{ij})$ stores the positive weights for node pairs. Rather than minimizing $\mathbf{c}^T\mathbf{x} = \sum_{i<j} w_{ij} m_{ij}$, we have a method that can minimize the quadratic objective $\mathbf{c}^T\mathbf{x} + \frac{1}{2\gamma}\mathbf{x}^T\mathbf{W}\mathbf{x}$ over the same constraint set. We construct a weight matrix that contains two copies of the weight vector $\mathbf{w}$, one to match up with the $\mathbf{y}$ vector and one corresponding to $\mathbf{m}$:

$$\mathbf{W} = \begin{bmatrix} diag(\mathbf{w}) & \mathbf{0} \\ \mathbf{0} & diag(\mathbf{w}) \end{bmatrix}. \tag{5.2}$$

The quadratic regularization of the original LP objective is then

$$\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} + \frac{1}{2\gamma}\mathbf{x}^T\mathbf{W}\mathbf{x} = \min_{(m_{ij}),(y_{ij})} \sum_{i<j} w_{ij} m_{ij} + \frac{1}{2\gamma}\sum_{i<j} w_{ij} m_{ij}^2 + \frac{1}{2\gamma}\sum_{i<j} w_{ij} y_{ij}^2. \tag{5.3}$$

For both (5.1) and (5.3), the constraints enforce $|y_{ij}| \le m_{ij}$. Since the objectives are being minimized, this guarantees $m_{ij} = |y_{ij}| \implies m_{ij}^2 = y_{ij}^2$ at optimality. This allows us to replace y_{ij}^2 with m_{ij}^2 in (5.3). We rewrite the objective using terms only involving m_{ij} variables:

$$\min_{(m_{ij}),(y_{ij})} \sum_{i<j} w_{ij} m_{ij} + \frac{1}{\gamma}\sum_{i<j} w_{ij} m_{ij}^2. \tag{5.4}$$

Let (m_{ij}^*) and (y_{ij}^*) be optimal for (5.1) and $(\hat{m}_{ij}), (\hat{y}_{ij})$ be optimal for (5.4). Then

$$\sum_{i<j} w_{ij}\hat{m}_{ij} + \frac{1}{\gamma}\sum_{i<j} w_{ij}\hat{m}_{ij}^2 \le \sum_{i<j} w_{ij} m_{ij}^* + \frac{1}{\gamma}\sum_{i<j} w_{ij}(m_{ij}^*)^2 \le \left(1 + \frac{1}{\gamma}\right)\sum_{i<j} w_{ij} m_{ij}^*. \tag{5.5}$$

In the last step above we have used the fact that $m_{ij}^* = |y_{ij}^*| \le 1 \implies m_{ij}^* \le (m_{ij}^*)^2$. This proves an approximation result for correlation clustering.

Theorem 5.1. *Let (m^*_{ij}) and (y^*_{ij}) be the optimal solution vectors for the correlation clustering LP relaxation given in (5.1), and let $(\hat{m}_{ij}), (\hat{y}_{ij})$ be the optimal solution to (5.3). Then*

$$\textstyle\sum_{i<j} w_{ij} m^*_{ij} \leq \sum_{i<j} w_{ij} \hat{m}_{ij} \leq (1 + 1/\gamma) \sum_{i<j} w_{ij} m^*_{ij}.$$

Therefore, given any rounding procedure for the original LP that gives a factor p approximation for a correlation clustering problem, we can instead solve the related QP using projection methods and round the output to obtain a factor $p(1 + 1/\gamma)$ approximation. For weighted correlation clustering, the best rounding procedures guarantee an $O(\log n)$ approximation [13, 17, 22], so this can still be achieved even if we use a small value for γ.

5.2. Sparsest cut. The Leighton–Rao LP relaxation for sparsest cut is presented in (3.2). This LP has a variable x_{ij} for every pair of distinct nodes $i < j$ in some unweighted graph $G = (V, E)$. Let $\mathbf{x} = (x_{ij})$ be a linearization of these distance variables, and define $\mathbf{c} = (c_{ij})$ to be the adjacency indicators, i.e., $c_{ij} = 1$ if $(i, j) \in E$ and $c_{ij} = 0$ otherwise, so that the objective can be written as $\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x}$. If we assume we have chosen a weight matrix $\mathbf{W}$ and a parameter $\gamma > 0$, the regularized version of (3.2) has the objective

$$\min_{\mathbf{x}} \sum_{i<j} c_{ij} x_{ij} + \frac{1}{2\gamma} \sum_{i<j} w_{ij} x_{ij}^2.$$

If we set $w_{ij} = c_{ij}$, we would be able to use steps analogous to our approach for dense correlation clustering and obtain a $1/(2\gamma)$ approximation for the original LP, by solving the regularized QP. However, many of the c_{ij} variables are zero, and we must avoid setting $w_{ij} = 0$, since $\mathbf{W}$ needs to be positive definite in order for our projection method to apply. Instead we introduce another parameter, $\lambda \in (0, 1)$, and define a set of weights $\mathbf{w} = (w_{ij})$, where $w_{ij} = 1$ if $(i, j) \in E$ and $w_{ij} = \lambda$ otherwise. In this way, the weight w_{ij} is still positive but can be *near* zero (i.e., near c_{ij}) when $(i, j) \notin E$. We can then prove the following approximation result.

Theorem 5.2. *Let $G = (V, E)$ be a connected graph with $n = |V| > 4$. Let ϕ^* be the minimum cut sparsity for G and assume that each side of the sparsest cut partition has at least two nodes. Let $\gamma > 0$, $\mathbf{W} = diag(\mathbf{w})$ be defined as above for a given $\lambda \in (0, 1)$, and let $\mathcal{A}$ denote the set of constraints from the Leighton–Rao LP relaxation for sparsest cut. Then*

$$\min_{\mathbf{x} \in \mathcal{A}} \mathbf{c}^T \mathbf{x} + \frac{1}{2\gamma} \mathbf{x}^T \mathbf{W} \mathbf{x} \leq \left(1 + \frac{1 + \lambda n}{2\gamma}\right) \phi^*.$$

Proof. The quadratic regularization of the sparsest cut LP relaxation is

$$\begin{array}{lll} \text{minimize} & \sum_{i<j} c_{ij} x_{ij} + (1/2\gamma) \sum_{i<j} w_{ij} x_{ij}^2 & \\ \text{subject to} & \sum_{i<j} x_{ij} = n & \\ & x_{ij} \leq x_{ik} + x_{jk} & \text{for all } i, j, k, \\ & x_{ij} \geq 0 & \text{for all } i, j. \end{array} \tag{5.6}$$

The result we prove here relates the optimal solution of (5.6) directly back to the minimum sparsity ϕ^*, rather than back to the LP relaxation of sparsest cut (3.2). This makes sense given that our purpose in solving these convex relaxations is to develop approximation results for the original NP-hard sparsest cut objective.

Let $S^* \subset V$ be the set of nodes inducing the sparsest cut partition of G, so that

$$\phi^* = \frac{\mathbf{cut}(S^*)}{|S^*|} + \frac{\mathbf{cut}(S^*)}{|\bar{S}^*|} = \frac{n\mathbf{cut}(S^*)}{|S^*||\bar{S}^*|}.$$

Without loss of generality, assume $|S^*| \leq |\bar{S}^*|$. In the statement of the theorem we assume that G is connected, $n > 4$, and $|S^*| > 1$. The connectivity of G ensures the problem cannot be trivially solved by finding a single connected component, and guarantees that $\mathbf{cut}(S^*) \geq 1$. Together the remaining two assumptions guarantee that $\frac{n}{|S^*||\bar{S}^*|} \leq \frac{n}{2(n-2)} \leq 1$, which will be useful later in the proof. We will also use the fact that $\frac{n}{|S^*||\bar{S}^*|} \leq \frac{n\mathbf{cut}(S^*)}{|S^*||\bar{S}^*|} = \phi^*$. Note that if $n \leq 4$, the problem is trivial to solve by checking all possible partitions, and if $|S^*| = 1$, then the sparsest cut problem is easy to solve by checking all n partitions that put a single node by itself.

In order to encode the optimal partition as a vector, define $\mathbf{s}^* = (s^*_{ij})$ by

$$s^*_{ij} = \begin{cases} \frac{n}{|S^*||\bar{S}^*|} & \text{if nodes } i \text{ and } j \text{ are on opposite sides of the partition } \{S^*, \bar{S}^*\}, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that this vector $\mathbf{s}^*$ satisfies the constraints of (5.6) and that

$$\sum_{i<j} c_{ij} s^*_{ij} = \sum_{(i,j)\in E} s^*_{ij} = \frac{\mathbf{cut}(S^*)n}{|S^*||\bar{S}^*|} = \phi^*.$$

We can also prove a useful bound on the quadratic term in the objective:

$$\begin{aligned}(\mathbf{s}^*)^T \mathbf{W} \mathbf{s}^* &= \sum_{i<j} w_{ij}(s^*_{ij})^2 = \sum_{(i,j)\in E} (s^*_{ij})^2 + \sum_{(i,j)\notin E} \lambda (s^*_{ij})^2 \\ &< \sum_{(i,j)\in E} (s^*_{ij})^2 + \sum_{i<j} \lambda (s^*_{ij})^2 = \mathbf{cut}(S^*) \frac{n^2}{|S^*|^2|\bar{S}^*|^2} + \lambda |S^*||\bar{S}^*| \frac{n^2}{|S^*|^2|\bar{S}^*|^2} \\ &= \phi^* \frac{n}{|S^*||\bar{S}^*|} + \lambda n \frac{n}{|S^*||\bar{S}^*|} \leq \phi^*(1 + \lambda n),\end{aligned}$$

where we have used the fact that $n/(|S^*||\bar{S}^*|) \leq \min\{1, \phi^*\}$ because of our simple assumptions on G. Let $\hat{\mathbf{x}}$ be the optimal solution for QP (5.6), and recall that $\mathbf{s}^*$ is another feasible point. We combine the bounds shown above to prove the final result:

$$\sum_{i<j} c_{ij}\hat{x}_{ij} < \sum_{i<j} c_{ij}\hat{x}_{ij} + \frac{1}{2\gamma}\sum_{i<j} w_{ij}\hat{x}^2_{ij} \leq \sum_{i<j} c_{ij} s^*_{ij} + \frac{1}{2\gamma}\sum_{i<j} w_{ij}(s^*_{ij})^2 \leq \phi^* + \frac{1}{2\gamma}(1 + \lambda n)\phi^*.$$

This completes the proof. ∎

6. Improved a posteriori approximations. The approximation bounds in the previous section provide a priori insights for how to set parameters γ and $\mathbf{W}$ before running Dykstra's projection algorithm to solve relaxations of sparsest cut and correlation clustering. In this section we outline improved a posteriori guarantees that can be achieved once we have obtained a solution to the regularized LP.

6.1. A first strategy for improved bounds. Consider again the optimal solutions to the linear program and its regularized objective $\mathbf{x}^* = \operatorname{argmin}_{\mathcal{A}} \mathbf{c}^T\mathbf{x}$ and $\hat{\mathbf{x}} = \operatorname{argmin}_{\mathcal{A}} \mathbf{c}^T\mathbf{x} + (\mathbf{x}^T\mathbf{W}\mathbf{x})/(2\gamma)$, where $\mathcal{A} = \{\mathbf{x} \in \mathbb{R}^N : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is the set of feasible solutions. For both correlation clustering and sparsest cut, we have proven a sequence of inequalities of the form

$$\mathbf{c}^T\mathbf{x}^* \leq \mathbf{c}^T\hat{\mathbf{x}} \leq \mathbf{c}^T\hat{\mathbf{x}} + \frac{1}{2\gamma}\hat{\mathbf{x}}^T\mathbf{W}\hat{\mathbf{x}} \leq \mathbf{c}^T\mathbf{x}^* + \frac{1}{2\gamma}(\mathbf{x}^*)^T\mathbf{W}(\mathbf{x}^*) \leq (1+A)OPT,$$

where A is a term in the approximation factor (in our settings, $1/\gamma$ or $(1+\lambda n)/\gamma$) and OPT is the optimal score for the NP-hard objective. Given $\hat{\mathbf{x}}$, we can improve this approximation result by computing $R = (\hat{\mathbf{x}}^T\mathbf{W}\hat{\mathbf{x}})/(2\gamma\mathbf{c}^T\hat{\mathbf{x}})$ and noting that

$$\mathbf{c}^T\hat{\mathbf{x}} + \frac{1}{2\gamma}\hat{\mathbf{x}}^T\mathbf{W}\hat{\mathbf{x}} = (1+R)\mathbf{c}^T\hat{\mathbf{x}} \implies \mathbf{c}^T\hat{\mathbf{x}} \leq \left(\frac{1+A}{1+R}\right)OPT.$$

We find that in a number of correlation clustering settings, computing R will provide a significantly improved approximation guarantee.

6.2. Improved guarantees by solving a small LP. We outline one more approach for getting improved approximation guarantees, this time based on a careful consideration of dual variables $\hat{\mathbf{y}}$ computed by Dykstra's method. We again start by considering the initial linear program which we assume is too challenging to solve using traditional LP software because of memory constraints,

$$\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} \quad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{6.1}$$

and let $\mathbf{x}^*$ denote the (unknown) optimizer for this LP. In practice, we solve a quadratic regularization:

$$\min_{\mathbf{x}} \ Q(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2\gamma}\mathbf{x}^T\mathbf{W}\mathbf{x} \quad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}. \tag{6.2}$$

We solve (6.2) to find a primal-dual pair of vectors $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ satisfying KKT conditions, and in particular, in section 4.3 we saw that

$$\frac{1}{\gamma}\mathbf{W}\hat{\mathbf{x}} = -\mathbf{A}^T\hat{\mathbf{y}} - \mathbf{c}, \tag{6.3}$$

$$-\mathbf{b}^T\hat{\mathbf{y}} - \frac{1}{2\gamma}\hat{\mathbf{x}}^T\mathbf{W}\hat{\mathbf{x}} = \mathbf{c}^T\hat{\mathbf{x}} + \frac{1}{2\gamma}\hat{\mathbf{x}}^T\mathbf{W}\hat{\mathbf{x}}. \tag{6.4}$$

Given this setup, we prove a new theorem for obtaining a lower bound on $\mathbf{c}^T\mathbf{x}^*$ by considering $\hat{\mathbf{y}}$ and solving another small, less computationally expensive LP.

Theorem 6.1. *Given $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, set $\hat{\mathbf{p}} = (1/\gamma)\mathbf{W}\hat{\mathbf{x}}$, and let $\tilde{\mathbf{x}}$ be the optimal solution to the following new linear program: $\max_{\mathbf{x}} \hat{\mathbf{p}}^T\mathbf{x}$ subject to $\mathbf{c}^T\mathbf{x} \leq \mathbf{c}^T\hat{\mathbf{x}}$ and $\mathbf{x} \in \mathcal{B}$, where $\mathcal{B}$ is any set which is guaranteed to contain $\mathbf{x}^*$ (i.e., $\mathcal{B}$ encodes a subset of constraints that are known to be satisfied by $\mathbf{x}^*$). Then we have the following lower bound on the optimal solution to (6.1):*

$$-\mathbf{b}^T\hat{\mathbf{y}} - \hat{\mathbf{p}}^T\tilde{\mathbf{x}} \leq \mathbf{c}^T\mathbf{x}^*. \tag{6.5}$$

Furthermore, if $\mathbf{x}^* = \hat{\mathbf{x}} = \tilde{\mathbf{x}}$, then this bound is tight.

Proof. The dual of the original linear program (6.1) is

$$\max \ -\mathbf{b}^T\mathbf{y} \ \text{ subject to } \ -\mathbf{A}^T\mathbf{y} - \mathbf{c} = 0 \text{ and } \ \mathbf{y} \geq 0. \tag{6.6}$$

One way to obtain a lower bound on $\mathbf{c}^T\mathbf{x}^*$ would be to find some feasible point $\mathbf{y}$ for (6.6), in which case $-\mathbf{b}^T\mathbf{y} \leq \mathbf{c}^T\mathbf{x}^*$. Note from (6.3) that we have access to a vector $\hat{\mathbf{y}}$ satisfying $\hat{\mathbf{y}} \geq 0$ and $-\mathbf{A}^T\hat{\mathbf{y}} - \mathbf{c} = \hat{\mathbf{p}} = (1/\gamma)\mathbf{W}\hat{\mathbf{x}}$. This $\hat{\mathbf{y}}$ is not feasible for (6.6), but we note that if the entries of $\hat{\mathbf{p}}$ are very small (which they will be for large γ), then the constraint $-\mathbf{A}^T\mathbf{y} - \mathbf{c} = 0$ is *nearly* satisfied by $\hat{\mathbf{y}}$. If we define a new vector $\hat{\mathbf{c}} = \mathbf{c} + \hat{\mathbf{p}}$, then we can observe that $\hat{\mathbf{y}}$ is feasible for a slightly perturbed linear program:

$$\max \ -\mathbf{b}^T\mathbf{y} \ \text{ subject to } \ -\mathbf{A}^T\mathbf{y} - \hat{\mathbf{c}} = 0 \text{ and } \ \mathbf{y} \geq 0. \tag{6.7}$$

Observe that this is the dual of a slight perturbation of the original LP (6.1):

$$\min_{\mathbf{x}} \ \hat{\mathbf{c}}^T\mathbf{x} \ \text{ subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}. \tag{6.8}$$

Since $\mathbf{x}^*$ is feasible for (6.8) and $\hat{\mathbf{y}}$ is feasible for its dual (6.7), we have the following inequality:

$$-\mathbf{b}^T\hat{\mathbf{y}} \leq \hat{\mathbf{c}}^T\mathbf{x}^* = \mathbf{c}^T\mathbf{x}^* + \hat{\mathbf{p}}^T\mathbf{x}^*. \tag{6.9}$$

Finally, observe that $\mathbf{x}^*$ is feasible for the LP (6.1) defined in the statement of the theorem, and therefore, $\hat{\mathbf{p}}^T\mathbf{x}^* \leq \hat{\mathbf{p}}^T\tilde{\mathbf{x}}$. Combining this fact with (6.9) we get our final result:

$$-\mathbf{b}^T\hat{\mathbf{y}} \leq \mathbf{c}^T\mathbf{x}^* + \hat{\mathbf{p}}^T\mathbf{x}^* \leq \mathbf{c}^T\mathbf{x}^* + \hat{\mathbf{p}}^T\tilde{\mathbf{x}} \implies -\mathbf{b}^T\hat{\mathbf{y}} - \hat{\mathbf{p}}^T\tilde{\mathbf{x}} \leq \mathbf{c}^T\mathbf{x}^*.$$

If we happen to choose $\gamma > 0$ and $\mathbf{W}$ in such a way that $\mathbf{x}^* = \hat{\mathbf{x}}$, and then pick a set $\mathcal{B}$ so that $\tilde{\mathbf{x}} = \mathbf{x}^*$, then property (6.4) ensures that this bound will be tight. ∎

6.3. A bound for sparsest cut. Consider the quadratic regularization of the sparsest cut relaxation shown in (5.6), with diagonal weight matrix defined as in section 5.2. Assume $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is the set of primal and dual variables obtained by solving the objective with Dykstra's method. We give a corollary to Theorem 6.1 that shows how to obtain good a posteriori approximations for how close $\mathbf{c}^T\hat{\mathbf{x}}$ is to the original LP relaxation of sparsest cut (3.2).

Corollary 6.2. *Let $\tilde{\mathbf{x}} = (\tilde{x}_{ij})$ be the optimizer for the following LP:*

$$\begin{array}{ll} \textit{maximize} & (1/\gamma)\sum_{i<j}(w_{ij}\hat{x}_{ij})x_{ij} \\ \textit{subject to} & \sum_{i<j} x_{ij} = n, \\ & \sum_{(i,j)\in E} x_{ij} \leq \sum_{(i,j)\in E} \hat{x}_{ij}, \\ & 0 \leq x_{ij} \leq n/(n-1) \qquad \textit{for all } i,j. \end{array} \tag{6.10}$$

Let $\hat{y}_1$ and $\hat{y}_2$ be correction variables within the dual vector $\hat{\mathbf{y}}$, corresponding to the constraints $\sum_{i<j} x_{ij} \leq n$ and $-\sum_{i<j} x_{ij} \leq -n$, respectively, which combine to form the equality constraint $\sum_{i<j} x_{ij} = n$. Then

$$-n\hat{y}_1 + n\hat{y}_2 - \frac{1}{\gamma}\sum_{i<j} w_{ij}\hat{x}_{ij}\tilde{x}_{ij} \leq \sum_{(i,j)\in E} x^*_{ij}\,.$$

Proof. We just need to show that the assumptions of Theorem 6.1 are satisfied. Let x^*_{ij} be the optimal solution vector for the sparsest cut LP relaxation (3.2). Note that $x^*_{ij} \leq n/(n-1)$ for all i, j. If this were not the case and $x^*_{uv} > n/(n-1)$ for some pair (u, v), then there would exist $(n-2)$ nodes k distinct from u and v such that $n/(n-1) < x^*_{uv} \leq x^*_{uk} + x^*_{vk}$. Then

$$\sum_{i<j} x^*_{ij} \geq x^*_{uv} + \sum_{u \neq k \neq v} x^*_{uk} + x^*_{vk} > \frac{n}{n-1} + (n-2)\frac{n}{n-1} = n,$$

which contradicts the fact that the entries of $\mathbf{x}^*$ sum to n. We see then that all of the constraints included in LP (6.10) are satisfied by x^*_{ij}. Note that the only nonzero terms in $\mathbf{b}$ are n and $-n$, which correspond to $\hat{y}_1$ and $\hat{y}_2$ in $\hat{\mathbf{y}}$. Therefore, $-\mathbf{b}^T\hat{\mathbf{y}} = -n\hat{y}_1 + n\hat{y}_2$. ∎

7. Experiments. We implement Dykstra-based solvers for relaxations of sparsest cut (DykstraSC) and correlation clustering (DykstraCC) in the Julia programming language. We additionally show how to apply DykstraCC to obtain good empirical results for the modularity objective. In practice, we solve sparsest cut and modularity relaxations on graphs with up to 3086 nodes, and dense correlation clustering problems with up to 11204 nodes. Thus, we solve optimization problems with up to 63 million edges and 7.0×10^{11} constraints. For correlation clustering, the previous best approaches managed to optimally solve instances with 13 million constraints [38], or solve a different, but related, LP relaxation on problems with 5 million edges [44]. For the metric nearness problem, Dhillon, Sra, and Tropp apply their triangle-fixing algorithm to solve metric nearness problems on random $n \times n$ dissimilarity matrices with n up to 5000 [19]. However, their method simply runs Dykstra's method until the change in the solution vector falls below a certain threshold. Since this approach does not take constraint satisfaction or duality gap into consideration, it comes with no output guarantees.

7.1. Implementation details for convergence check. Both DykstraSC and DykstraCC use the detailed convergence check outlined in section 4. To check whether $\tau > \rho_k$ for a constraint tolerance τ, after every pass through the constraints using Dykstra's method, our algorithm iterates again through the constraints and returns false as soon as it encounters a violated constraint, if one exists. Thus in early stages, the method can confirm that $\rho_k \geq \tau$ without visiting all $\Theta(n^3)$ constraints each time. After every C iterations, the algorithm performs a full pass through constraints to check the maximum violation ρ_k, and to apply the entrywise rounding procedure. For the rounding procedure, we set a preliminary threshold τ_0. After every C iterations, if $\rho_k < \tau_0$, the algorithm computes the rounded vector $\mathbf{x}_r$ for a range of r values. For sparsest cut, $\tau_0 = 0.1$ and $C = 10$, and we test each value of r from 2 to 6. In practice, the rounding procedure significantly increases the method's performance in finding solutions with constraints satisfied to within machine precision. For correlation clustering we focus only on solving relaxations to within an overall constraint tolerance of 0.01 (or 0.001 for related modularity experiments), and we do not see any performance gains using the rounding procedure. Regardless, by design, the rounding procedure neither dominates the runtime nor affects the method's ability to converge using the standard Dykstra iterate. In our weighted correlation clustering experiments, we perform the rounding procedure every $C = 20$ steps, leading to an overall runtime increase of around 5–7%. For modularity clustering, we check every $C = 10$ steps, leading to an increase between 10–15%.

7.2. Using Gurobi software. In our experiments our aim is obtain high-quality solutions to metric-constrained relaxations of graph clustering problems. Thus, we compare primarily against solving the same correlation clustering and sparsest cut relaxations using commercial Gurobi software. Gurobi possesses a number of solvers for LPs. In practice, we separately run Gurobi's barrier method (i.e., the interior-point solver), the primal simplex method, and the dual simplex method. For the interior-point method, Gurobi's default setting is to convert any solution it finds to a basic feasible solution, but we turn this setting off since we do not require this of our own solver and we are simply interested in finding any solution to the LP. In practice, we find that the interior-point solver is the fastest.

For both sparsest cut and correlation clustering we also test an additional *lazy-constraint* method when employing Gurobi software. This procedure solves the LP objective over a subset of metric constraints, checks for violated constraints, and then expands the constraint set and re-solves the problem until optimality. This procedure is especially helpful for the correlation clustering objective, though for the sparsest cut objective we find most metric constraints are tight at optimality, so there is little to no benefit to repeatedly solving the problem on growing subsets of constraints.

7.3. Real-world graphs. In our experiments we use real-world networks obtained almost exclusively from the SuiteSparse Matrix Collection [15] and the SNAP repository [36], with one graph (Vassar85) from the Facebook100 datasets. The graphs we experiment on come from numerous domains, including citation networks (SmallW and SmaGri), collaboration networks (caGrQc, caHepTh, caHepPh, Netscience, Erdos991), web-based graphs (Harvard500, Polblogs, Email, Vassar95), biological networks (C. El-Neural, C. Elegan), and others (Power, USAir97, Roget). Before running experiments, we make all edges undirected, remove edge weights, and find the largest connected component to ensure we are always working with connected, unweighted, and undirected networks.

7.4. The sparsest cut relaxation. We run DykstraSC on graphs ranging in size from 198 to 3068 nodes. Our machine has two 14-core 2.66 GHz Xeon processors and for ease of reproducibility we limit experiments to 100GB of RAM. Results are shown in Table 1. We also display runtimes and convergence plots in Figure 1. Gurobi has an advantage on smaller graphs, but slows down and then runs out of memory once the graphs scale beyond a few hundred nodes. Since DykstraSC is, in fact, optimizing a quadratic regularization of the sparsest cut LP relaxation, we also report how close our solution is to the optimal LP solution, either by comparing against Gurobi or using our a posteriori approximation guarantee, presented in Corollary 6.2. In nearly all cases we are within 1% of the optimal LP solution, and in several cases our solver returns the optimal LP solution.

The fastest results for Gurobi are obtained by running the interior-point solver. Gurobi runs out of memory when trying to form the entire constraint matrix for problems with more than 500 nodes, since, when $n = 500$, the matrix contains over 186 million nonzero entries. We manage to solve the relaxation on Harvard500 using the lazy constraints approach, though repeatedly solving subproblems on subsets of metric constraints leads to a very long runtime. For graphs larger than Harvard500, we run out of memory even when trying this approach.

Table 1

We solve the LP relaxation for sparsest cut via DykstraSC on 13 *graphs. For all datasets but the largest we set* $\gamma = 5$ *and* $\lambda = 1/n$*. For Vassar*85*, the largest and hence most expensive problem, we use* $\gamma = 2$ *and* $\lambda = 1/1000$*. These parameter settings lead to a faster convergence, at the expense of a slightly worse approximation guarantee. Both DykstraSC and Gurobi (when it doesn't run out of memory) solve the problems to within a relative gap tolerance of* 10^{-4}*, and satisfy constraints to within machine precision. The last column reports the ratio* Δ *between the LP objective output by our DykstraSC and the minimum LP score. In cases where we do not know* Δ *exactly, we report an upper bound computed using our a posteriori approximation guarantees (see section* 6.3*). Time is given in seconds.*

Graph	$\lvert V\rvert$	$\lvert E\rvert$	# constraints	Gurobi time	Dykstra time	Δ Approx
Jazz	198	2742	3.8×10^6	60	81	1.003
SmallW	233	994	6.2×10^6	93	166	1.001
C.El-Neural	297	2148	1.2×10^7	274	350	1.000
USAir97	332	2126	1.8×10^7	471	511	1.041
Netscience	379	914	2.7×10^7	887	1134	1.000
Erdos991	446	1413	4.4×10^7	2574	1954	1.011
C.El-Meta	453	2025	4.6×10^7	2497	1138	1.000
Harvard500	500	2043	6.2×10^7	18769	1427	1.000
Roget	994	3640	4.9×10^8	out of memory	53449	≤ 1.008
SmaGri	1024	4916	5.4×10^8	out of memory	25703	≤ 1.002
Email	1133	5451	7.3×10^8	out of memory	34621	≤ 1.005
Polblogs	1222	16714	9.1×10^8	out of memory	41080	≤ 1.013
Vassar85	3068	119161	1.4×10^{10}	out of memory	155333	≤ 1.165

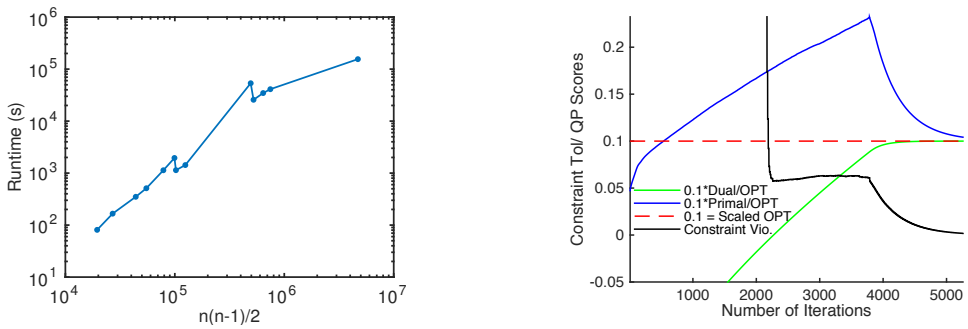

Figure 1. *On the left we show runtimes for DykstraSC on real-world graphs with* 198 *to* 3068 *nodes. If* n *is the number of nodes in the graph, then DykstraSC solves for* $n(n-1)/2$ *distance scores. We show a convergence plot specifically for the Polblogs dataset on the right. The red dotted line indicates the optimal quadratic objective, normalized to equal* 0.1*. The (normalized) dual objective (green) always gives a lower bound on the (normalized) optimal solution, whereas the (normalized) primal objective (blue) is not an upper bound for the first* ≈ 500 *iterations. We display the maximum constraint violation in black.*

7.5. Weighted correlation clustering. We convert several real-world graphs into instances of correlation clustering using the approach of Wang et al. [50]. The procedure is as follows:

1. Given $G = (V, E)$, compute the Jaccard coefficient between each pair of nodes i, j: $J_{ij} = |N(i) \cap N(j)|/|N(i) \cup N(j)|$, where $N(u)$ is the set of nodes adjacent to node u.
2. Apply a nonlinear function on Jaccard coefficients to obtain a score indicating similarity or dissimilarity: $S_{ij} = \log\left((1 + J_{ij} - \delta)/(1 - J_{ij} + \delta)\right)$. Here, δ is set so that $S_{ij} > 0$ if $J_{ij} > \delta$ and $S_{ij} < 0$ when $J_{ij} < \delta$. Following Wang et al. [50], we fix

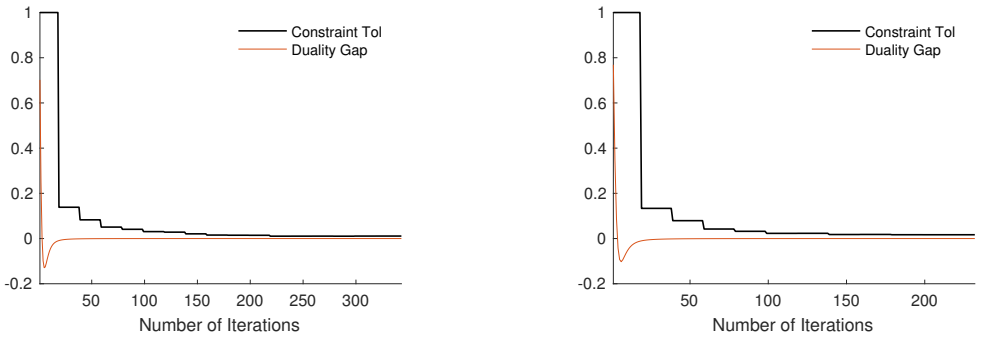

Figure 2. *We show convergence plots for DykstraCC on caHepTh (left) and caGrQc (right). In the first several iterations the duality gap* $(Q(\mathbf{x}) - D(\mathbf{y}))/D(\mathbf{y})$ *is negative, since Dykstra's method does not guarantee the primal scores* $Q(\mathbf{x})$ *will be greater than dual scores* $D(\mathbf{y})$ *at the beginning of the algorithm. After a few iterations, the maximum constraint violation (black) decreases significantly and the duality gap steadily goes to zero. In practice our solver recomputes the maximum constraint violation every* 20 *iterations, leading to jumps in the constraint violation curves displayed.*

$\delta = 0.05$.

3. Wang et al. stop after the above step and use S_{ij} scores for their correlation clustering problems. We additionally offset each entry by $\pm\epsilon$ to avoid cases where edge weights are zero:

$$Z_{ij} = \begin{cases} S_{ij} + \epsilon & \text{if } S_{ij} > 0, \\ S_{ij} - \epsilon & \text{if } S_{ij} < 0, \\ \epsilon & \text{if } S_{ij} = 0 \text{ and } (i,j) \in E, \\ -\epsilon & \text{if } S_{ij} = 0 \text{ and } (i,j) \notin E. \end{cases}$$

In the above construction, $S_{ij} = 0$ indicates there is no strong similarity or dissimilarity between nodes based on their Jaccard coefficient. If in this case nodes i and j are adjacent, we interpret this as a small indication of similarity and assign them a small positive weight. Otherwise, we assign a small negative weight. In all our experiments we fix $\epsilon = 0.01$. The sign of Z_{ij} indicates whether nodes i and j are similar or dissimilar, and $w_{ij} = |Z_{ij}| > 0$ is the nonnegative weight for the associated correlation clustering problem. Results for running DykstraCC and Gurobi on the resulting signed graphs are shown in Table 2. We show convergence plots for the two of the graphs in Figure 2.

On problems of this size, we restrict ourselves to using the lazy-constraint approach, coupled with Gurobi's interior-point solver. In one case, the lazy-constraint method converges very quickly. Effectively, it finds a small subset of constraints that are sufficient to force all other metric constraints to be satisfied at optimality. However, Gurobi runs out of memory on the other large problems considered, indicating that, even if we are extremely careful, standard off-the-shelf solvers are unable to compete with our Dykstra-based approach.

Because the correlation clustering problems we address are so large, we set $\gamma = 1$ and run Dykstra's method until constraints are satisfied to within a tolerance of 0.01. We find that long before the constraint tolerance reaches this point, the duality gap shrinks below 10^{-4}. We note that although it takes a long time to reach convergence on graphs with thousands of nodes, DykstraCC has no issues with memory. Monitoring the memory usage of our machine,

Table 2

DykstraCC solves convex relaxations of correlation clustering with up to 700 billion constraints. The lazy-constraint Gurobi solver does very well for one very sparse graph, but runs out of memory on all other problems. We set $\gamma = 1$, and constraint tolerance to 0.01. Selecting a small γ leads to poorer approximation guarantees, but dramatically decreases the number of needed iterations until convergence. For problems on which we cannot optimally solve the LP with Gurobi to obtain an approximation ratio Δ, we report an upper bound.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	# constraints	Gurobi time	Dykstra time	Δ Approx
power	4941	6594	6.0×10^{10}	549 s	7.6 hrs	1.07
caGrQc	4158	13422	3.6×10^{10}	out of memory	6.6 hrs	≤ 1.33
caHepTh	8638	24806	3.2×10^{11}	out of memory	88.3 hrs	≤ 1.34
caHepPh	11204	117619	7.0×10^{11}	out of memory	173.7 hrs	≤ 1.27

we noted that for the 11204 node graph, DykstraCC was using only around 12GB of the 100GB of available RAM. Given enough time, therefore, we expect our method to be able to solve metric-constrained LPs on a much larger scale. The ability to solve these relaxations on problems of this scale is already an accomplishment, given the fact that standard optimization software often fails on graphs with even a few hundred nodes.

7.6. Maximum modularity clustering via LP rounding. For our last experiment we use DykstraCC to obtain approximations to the popular maximum modularity graph clustering objective [39]. Although modularity is NP-hard to approximate to within any constant factor [20], solving its LP relaxation allows practitioners to obtain good a posteriori guarantees for fast heuristics such as the celebrated Louvain method [11]. Previous work in this area managed to solve the metric-constrained relaxation of modularity on graphs with up to 235 nodes [1]. Later, Aloise et al. [3] developed an approach for exactly solving modularity which succeeded on graphs with up to 512 nodes. With our approach we solve relaxations with thousands of nodes, and can quickly obtain good bounds on modularity for smaller graphs. Additionally, we demonstrate that rounding the LP and then greedily improving the output with the Louvain algorithm often leads to clusterings with higher modularity than running Louvain by itself.

Modularity objective. The maximum modularity objective for a graph $G = (V, E)$ is

$$\max_{\mathcal{C}} \ \mathbf{M}(\mathcal{C}) = \frac{1}{2|E|} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2|E|} \right) \delta_{ij}^{\mathcal{C}}, \tag{7.1}$$

where d_i is the degree of node i, and A_{ij} is the $\{0, 1\}$ indicator for whether i, j are adjacent in G. The $\delta_{ij}^{\mathcal{C}}$ variables encode the clustering, i.e., $\delta_{ij}^{\mathcal{C}} = 1$ if i, j are together in $\mathcal{C}$ and $\delta_{ij}^{\mathcal{C}} = 0$ otherwise. In previous work we showed that the modularity objective is a linear function of a specially weighted correlation clustering objective called LambdaCC [48]. Specifically, if we are given an input graph $G = (V, E)$ on which we wish to perform maximum modularity clustering, we construct from it an instance of correlation clustering in which an edge $(i, j) \in E$ is mapped to a positive edge of weight $1 - d_i d_j/(2|E|)$ in a new signed graph, and $(i, j) \notin E$ is mapped to a negative edge with weight $d_i d_j/(2|E|)$. The modularity objective $\mathbf{M}$ for a clustering $\mathcal{C}$ and the corresponding correlation clustering objective $\mathbf{CC}$ are then related as

follows:

$$\mathbf{CC}(\mathcal{C}) = m(1 - \mathbf{M}(\mathcal{C})) - \sum_{(i,j)\in E} \frac{d_i d_j}{2m} + \sum_{i=1}^{n} \frac{d_i^2}{4m}, \tag{7.2}$$

where $m = |E|$ and $n = |V|$. We note that this relationship also holds for relaxed clusterings, i.e., we can replace $\delta_{ij}^{\mathcal{C}}$ in (7.1) with $(1 - x_{ij})$ where x_{ij} are relaxed distance variables satisfying metric constraints. One way to upper bound the maximum value of $\mathbf{M}$ is to first approximately minimize the related instance of correlation clustering using our projection method, and then compute approximation bounds for the correlation clustering objective using guarantees in sections 5 and 6. Let $\mathcal{C}^*$ be the optimal clustering for modularity and correlation clustering and $\tilde{\mathcal{C}}$ represent the *relaxed* clustering output by DykstraCC. If we know $\mathbf{CC}(\tilde{\mathcal{C}}) \leq (1+\delta)\mathbf{CC}(\mathcal{C}^*)$ for some $\delta > 0$, then using the relationship in (7.2), we can upper bound the maximum modularity:

$$\mathbf{M}(\mathcal{C}^*) \leq \frac{\mathbf{M}(\tilde{\mathcal{C}})}{1+\delta} + \frac{\delta}{1+\delta}\left(1 - \sum_{(i,j)\in E} \frac{d_i d_j}{2m^2} + \sum_{i=1}^{n} \frac{d_i^2}{(2m)^2}\right). \tag{7.3}$$

Modularity scores range between -1 and 1 for any fixed clustering, and for sufficiently small δ the additive approximation above will provide a good upper bound.

We run DykstraCC on a subset of the larger graphs from the first experiment. The weighted correlation clustering problem we are solving here is an instance of LambdaCC with a small resolution parameter $\lambda = 1/(2|E|)$. In our previous work we showed that for small resolution parameters, LambdaCC is closely related to the sparsest cut and normalized cut objectives [48]. In practice, we are not surprised to find, therefore, that applying our solver to this problem is more similar to our experiments on sparsest cut than the weighted correlation clustering problems in the previous section. We find that many triangle constraints are tight and there is a significant memory requirement even for problems with just a few thousand nodes. Nevertheless, our approach scales to problems an order of magnitude larger than previous results.

LP rounding and Louvain. We compute the LP relaxation to within a constraint tolerance of 10^{-3} and a duality gap of 10^{-4}. We then round the (x_{ij}) variables into clusterings using our pivoting technique ThreeLP [48]. The method repeatedly selects a uniform random node, clusters it with all its neighbors within LP-distance 1/3, then removes the cluster and recurses on the rest of the graph. The method provides no a priori guarantees for $\mathbf{M}(\mathcal{C}^*)$ or $\mathbf{CC}(\mathcal{C}^*)$, but is a natural approach to test as it has provable approximation guarantees for a certain variant of LambdaCC when the resolution parameter is larger than 1/2. The rounding scheme is very fast, so we take the best of 50 instantiations each time we run it.

The Louvain method is a very popular heuristic developed by Blondel et al. [11] for maximizing modularity. The method takes an input clustering and repeatedly performs agglomerative moves that greedily improve the objective. We test Louvain in two ways: we first apply the method on the input clustering in which each node belongs to its own cluster, which is the standard initialization for Louvain. We then run Louvain as a way to greedily improve and refine the clustering output by the LP rounding procedure. In practice, we use

Table 3

We approximate the LP relaxation of modularity using our projection methods with $\gamma = 2$. Runtime for solving the relaxation is given in column 3. We compute an upper bound (UB) on the maximum modularity using (7.3). We obtain clusterings with the Louvain method (Louv), a simple LP rounding technique (3LP), and by using the output of 3LP as input to Louvain. We report maximum/median modularity scores over 15 trials. Shown in bold are median and maximum scores that are higher than those achieved with only Louvain.

Graph	n	Time	UB	Louv	3LP	3LP+Lou
Netscience	379	56s	0.8652	0.8484/0.8417	0.8310/0.8276	**0.8486**/**0.8485**
C.El-Meta	453	146s	0.5479	0.4478/0.4421	0.2640/0.2479	**0.4517**/**0.4485**
Erdos991	446	167s	0.6602	0.5374/0.5293	0.3498/0.3441	**0.5391**/**0.5369**
Harvard500	500	167s	0.7630	0.7386/0.7349	0.6868/0.6817	0.7386/**0.7386**
Roget	994	2189s	0.7304	0.5438/0.5383	0.2938/0.2888	**0.5472**/**0.5434**
SmaGri	1024	1897s	0.6124	0.4755/0.4697	0.2137/0.2066	**0.4773**/**0.4757**
Email	1133	3203s	0.6880	0.5788/0.5766	0.3356/0.3240	**0.5811**/**0.5789**
Polblogs	1222	3638s	0.5170	0.4270/0.4268	0.1658/0.1300	0.4270/**0.4270**
Vassar85	3068	48.2hr	0.5641	0.3958/0.3957	0.0950/0.0940	0.3958/0.3957

the Louvain software of Jeub et al. [30], which includes randomized variations that can lead to higher-modularity outputs. In Table 3 we report the best modularity and median modularity returned over 15 runs for each approach. We observe that LP rounding on its own is not competitive with Louvain. This is not surprising given that the rounding scheme performs simplistic clustering moves that are easy to analyze, but are less sophisticated and intelligent than the Louvain heuristics. However, we notice that combining LP rounding with the Louvain method leads to a more robust approach with higher median and maximum scores. While the improvement is only slight, it is consistent. This indicates that solving the LP relaxation is useful not only for providing bounds on NP-hard objectives, but can also be used as a guide for making heuristic algorithms more robust. This suggests future work in developing LP rounding techniques that are specifically designed to initialize greedy local heuristics like Louvain using global knowledge about the problem instance.

8. Discussion and future challenges. We have developed an approach for solving expensive convex relaxations of clustering objectives that works on a much larger scale than was previously possible. We now observe several challenges that seem inherent in improving this approach, without significantly departing from the application of projection methods. We tried variants of Bauschke's method [7] and Haugazeau's projection method [25, 8], which do not compute dual variables as Dykstra's does. Such methods hence require only $O(n^2)$ memory, instead of $O(n^3)$, but come with significantly worse convergence guarantees. Because it is hard to determine, in practice, when these methods have converged, it is difficult to use them to obtain an output satisfying any guarantees with respect to the optimal solution. Other natural approaches to consider are the use of parallelization or randomization. Parallel versions of Dykstra's method exist [28], but they rely on averaging a large number of very tiny changes in each iteration, equal to the number of constraints. Since in our case there are $O(n^3)$ constraints, we find that, in practice, this averaging approach leads to changes that are so small that no meaningful progress can be made from one iteration to the next. The challenge in using a randomized approach (see [29]) is that visiting constraints at random leads to a

much higher cost for visiting the same number of constraints. This is because accessing dual variables at random from a dictionary is slower in practice than sequentially visiting elements in an array of dual variables.

REFERENCES

[1] G. Agarwal and D. Kempe, *Modularity-maximizing graph communities via mathematical programming*, Eur. Phys. J. B, 66 (2008), pp. 409–418, https://doi.org/10.1140/epjb/e2008-00425-1.

[2] N. Ailon, M. Charikar, and A. Newman, *Aggregating inconsistent information: Ranking and clustering*, J. ACM, 55 (2008), 23.

[3] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti, and S. Perron, *Column generation algorithms for exact modularity maximization in networks*, Phys. Rev. E, 82 (2010), 046112, https://doi.org/10.1103/PhysRevE.82.046112.

[4] S. Arora, S. Rao, and U. Vazirani, *Expander flows, geometric embeddings and graph partitioning*, J. ACM, 56 (2009), 5.

[5] D. Avis and J. Umemoto, *Stronger linear programming relaxations of max-cut*, Math. Program., 97 (2003), pp. 451–469, https://doi.org/10.1007/s10107-003-0423-5.

[6] N. Bansal, A. Blum, and S. Chawla, *Correlation clustering*, Mach. Learn., 56 (2004), pp. 89–113.

[7] H. H. Bauschke, *The approximation of fixed points of compositions of nonexpansive mappings in Hilbert space*, J. Math. Anal. Appl., 202 (1996), pp. 150–159, https://doi.org/10.1006/jmaa.1996.0308.

[8] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, Cham, 2017.

[9] J. Berg and M. Järvisalo, *Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability*, Artificial Intelligence, 244 (2017), pp. 110–142.

[10] E. G. Birgin and M. Raydan, *Robust stopping criteria for Dykstra's algorithm*, SIAM J. Sci. Comput., 26 (2005), pp. 1405–1414, https://doi.org/10.1137/03060062X.

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, J. Stat. Mech., 2008 (2008), P10008, https://doi.org/10.1088/1742-5468/2008/10/P10008.

[12] J. Brickell, I. S. Dhillon, S. Sra, and J. A. Tropp, *The metric nearness problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 375–396, https://doi.org/10.1137/060653391.

[13] M. Charikar, V. Guruswami, and A. Wirth, *Clustering with qualitative information*, J. Comput. System Sci., 71 (2005), pp. 360–383, https://doi.org/10.1016/j.jcss.2004.10.012.

[14] S. Chawla, K. Makarychev, T. Schramm, and G. Yaroslavtsev, *Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs*, in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, ACM, New York, 2015, pp. 219–228.

[15] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Softw., 38 (2011), 1, https://doi.org/10.1145/2049662.2049663.

[16] A. Dax, *The adventures of a simple algorithm*, Linear Algebra Appl., 361 (2003), pp. 41–61, https://doi.org/10.1016/S0024-3795(01)00600-0.

[17] E. D. Demaine and N. Immorlica, *Correlation clustering with partial information*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, S. Arora, K. Jansen, J. D. P. Rolim, and A. Sahai, eds., Lecture Notes in Comput. Sci. 2764, Springer, Berlin, 2003, pp. 1–13.

[18] I. S. Dhillon, S. Sra, and J. A. Tropp, *The Metric Nearness Problems with Applications*, Department of Computer Sciences, Technical Report TR-03-23, The University of Texas at Austin, Austin, TX, 2003.

[19] I. S. Dhillon, S. Sra, and J. A. Tropp, *Triangle fixing algorithms for the metric nearness problem*, in Proceedings of the 17th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, 2004, pp. 361–368, http://dl.acm.org/citation.cfm?id=2976040.2976086.

[20] T. N. Dinh, X. Li, and M. T. Thai, *Network clustering via maximizing modularity: Approximation algorithms and theoretical limits*, in Proceedings of the 2015 IEEE International Conference on Data Mining, ICDM 2015, IEEE, Washington, DC, 2015, pp. 101–110.

[21] R. L. Dykstra, *An algorithm for restricted least squares regression*, J. Amer. Statist. Assoc., 78 (1983), pp. 837–842.

[22] D. Emanuel and A. Fiat, *Correlation clustering – Minimizing disagreements on arbitrary weighted graphs*, in European Symposium on Algorithms, Springer, Berlin, Heidelberg, 2003, pp. 208–220.

[23] R. Escalante and M. Raydan, *Alternating Projection Methods*, SIAM, Philadelphia, 2011, https://doi.org/10.1137/9781611971941.

[24] D. Glasner, S. N. Vitaladevuni, and R. Basri, *Contour-based joint clustering of multiple segmentations*, in Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, IEEE Computer Society, Washington, DC, 2011, pp. 2385–2392, https://doi.org/10.1109/CVPR.2011.5995436.

[25] Y. Haugazeau, *Sur les inéquations variationnelles et la minimisation de fonctionnelles convexes*, Ph.D. thesis, Universite de Paris, Paris, France, 1968.

[26] C. Hildreth, *A quadratic programming procedure*, Naval Res. Logist. Quart., 4 (1957), pp. 79–85.

[27] J. P. Hou, A. Emad, G. J. Puleo, J. Ma, and O. Milenkovic, *A new correlation clustering method for cancer mutation analysis*, Bioinformatics, 32 (2016), pp. 3717–3728, https://doi.org/10.1093/bioinformatics/btw546.

[28] A. N. Iusem and A. R. De Pierro, *On the convergence of Han's method for convex programming with quadratic objective*, Math. Programming, 52 (1991), pp. 265–284.

[29] N. Jamil, X. Chen, and A. Cloninger, *Hildreth's algorithm with applications to soft constraints for user interface layout*, J. Comput. Appl. Math., 288 (2015), pp. 193–202, https://doi.org/https://doi.org/10.1016/j.cam.2015.04.014.

[30] L. G. S. Jeub, M. Bazzi, I. S. Jutla, and P. J. Mucha, *A Generalized Louvain Method for Community Detection Implemented in MATLAB,* 2011–2017, 2017, http://netwiki.amath.unc.edu/GenLouvain/.

[31] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo, *Higher-order correlation clustering for image segmentation*, in Proceedings of the 24th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, 2011, pp. 1530–1538; http://dl.acm.org/citation.cfm?id=2986459.2986630.

[32] K. Lang and S. Rao, *Finding near-optimal cuts: An empirical evaluation*, in Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1993, SIAM, Philadelphia, 1993, pp. 212–221.

[33] K. J. Lang, M. W. Mahoney, and L. Orecchia, *Empirical evaluation of graph partitioning using spectral embeddings and flow*, in International Symposium on Experimental Algorithms, SEA 2009, Springer, Berlin, Heidelberg, 2009, pp. 197–208.

[34] J.-H. Lange, A. Karrenbauer, and B. Andres, *Partial optimality and fast lower bounds for weighted correlation clustering*, in Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, 2018, PMLR, pp. 2892–2901; available online at http://proceedings.mlr.press/v80/lange18a.html.

[35] T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, J. ACM, 46 (1999), pp. 787–832.

[36] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*, http://snap.stanford.edu/data, June 2014.

[37] O. L. Mangasarian, *Normal solutions of linear programs*, in Mathematical Programming at Oberwolfach II, Math. Programming Stud. 22, Springer, Berlin, Heidelberg, 1984, pp. 206–216.

[38] A. Miyauchi, T. Sonobe, and N. Sukegawa, *Exact clustering via integer programming and maximum satisfiability*, in Thirty-Second AAAI Conference on Artificial Intelligence, 2018; available online at https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16176.

[39] M. E. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E, 69 (2004), 026113.

[40] S. Poljak and Z. Tuza, *The expected relative error of the polyhedral approximation of the max-cut problem*, Oper. Res. Lett., 16 (1994), pp. 191–198, https://doi.org/10.1016/0167-6377(94)90068-X.

[41] G. J. Puleo and O. Milenkovic, *Correlation clustering with constrained cluster sizes and extended weights bounds*, SIAM J. Optim., 25 (2015), pp. 1857–1872, https://doi.org/10.1137/140994198.

[42] G. J. Puleo and O. Milenkovic, *Correlation clustering and biclustering with locally bounded errors*, in Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML 2016, 2016, pp. 869–877, http://dl.acm.org/citation.cfm?id=3045390.3045483.

[43] C. Swamy, *Correlation clustering: Maximizing agreements via semidefinite programming*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, SIAM, Philadelphia, 2004, pp. 526–527.

[44] P. Swoboda and B. Andres, *A message passing algorithm for the minimum cost multicut problem*, in Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, IEEE, 2017, pp. 1617–1626.

[45] R. J. Tibshirani, *Dykstra's algorithm, ADMM, and coordinate descent: Connections, insights, and extensions*, in Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, 2017, pp. 517–528; http://dl.acm.org/citation.cfm?id=3294771.3294821.

[46] J. Van Gael and X. Zhu, *Correlation clustering for crosslingual link detection*, in Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI 2007, Morgan Kaufmann, San Francisco, 2007, pp. 1744–1749, http://dl.acm.org/citation.cfm?id=1625275.1625558.

[47] A. van Zuylen and D. P. Williamson, *Deterministic pivoting algorithms for constrained ranking and clustering problems*, Math. Oper. Res., 34 (2009), pp. 594–620, https://doi.org/10.1287/moor.1090.0385.

[48] N. Veldt, D. F. Gleich, and A. Wirth, *A correlation clustering framework for community detection*, in Proceedings of the 2018 World Wide Web Conference, 2018, pp. 439–448, https://doi.org/10.1145/3178876.3186110.

[49] S. N. Vitaladevuni and R. Basri, *Co-clustering of image segments using convex optimization applied to EM neuronal reconstruction*, in Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, IEEE, pp. 2203–2210, https://doi.org/10.1109/CVPR.2010.5539901.

[50] Y. Wang, L. Xu, Y. Chen, and H. Wang, *A scalable approach for general correlation clustering*, in International Conference on Advanced Data Mining and Applications, ADMA 2013, Springer, Berlin, Heidelberg, 2013, pp. 13–24.

[51] A. Wirth, *Approximation Algorithms for Clustering*, Ph.D. thesis, Princeton University, Princeton, NJ, 2004.

[52] A. Wirth, *Correlation clustering*, in Encyclopedia of Machine Learning, Springer, Boston, 2010, pp. 227–231, https://doi.org/10.1007/978-0-387-30164-8_176.