PROJECT AND FORGET: SOLVING LARGE-SCALE METRIC CONSTRAINED PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Given a set of distances amongst points, determining what metric representation is most "consistent" with the input distances or the metric that best captures the relevant geometric features of the data is a key step in many machine learning algorithms. In this paper, we focus on metric constrained problems, a class of optimization problems with *metric constraints*. In particular, we identify three types of metric constrained problems: metric nearness (Brickell et al. (2008)), weighted correlation clustering on general graphs (Bansal et al. (2004)), and metric learning (Bellet et al. (2013); Davis et al. (2007)). Because of the large number of constraints in these problems, however, these and other researchers have been forced to restrict either the kinds of metrics learned or the size of the problem that can be solved.

We provide an algorithm, PROJECT AND FORGET, that uses Bregman projections with cutting planes, to solve metric constrained problems with many (possibly exponentially) inequality constraints. We also prove that our algorithm converges to the global optimal solution. Additionally, we show that the optimality error (L_2 distance of the current iterate to the optimal) asymptotically decays at an exponential rate. We show that using our method we can solve large problem instances of three types of metric constrained problems, out-performing all state of the art methods with respect to CPU times and problem sizes.

1 INTRODUCTION

Given a set of distances amongst data points, many machine learning algorithms are considerably "easier" once these distances adhere to a metric. Furthermore, learning what metric is most "consistent" with the input distances or the metric that best captures the relevant geometric features of the data (e.g., the correlation structure in the data) is a key step in efficient, approximation algorithms for classification, clustering, regression, feature selection, etc. Indyk (1999) provides a list of other computational problems such as nearest neighbor search, (approximate) proximity problems, facility location, and a variety of graph problems for which we have efficient approximation algorithms in a general metric space. Given the importance of metric representations of data sets, we focus on metric constrained problems, a class of optimization problems with *metric constraints*; i.e., optimization of a convex function subject to metric constraints, such as the triangle inequality, on all the output variables.

In particular, we identify three types of metric constrained problems: *metric nearness* (Brickell et al. (2008)), *weighted correlation clustering on general graphs* (Bansal et al. (2004)), and *metric learning* (Bellet et al. (2013); Davis et al. (2007)). Briefly, the metric nearness problem seeks the closest metric to a given set of distances, the goal of correlation clustering is to partition nodes in a graph according to their similarity, and metric learning finds a metric on a dataset that is consistent with (dis)similarity information about the data points.

All of these problems can be modeled as constrained convex optimization problems with a large number of constraints. Unfortunately, because of the large number of constraints, using standard optimization techniques, researchers have been forced to restrict either the kinds of metrics learned or the size of the problem that can be solved.

Many of the existing methods for metric constrained problems suffer from some sort of significant drawback that hampers performance or restricts the instance size. Gradient based algorithms such as projected gradient descent (e.g., Beck & Teboulle (2009); Nesterov (1983)) or Riemannian gradient descent require a projection onto the space of all metrics, which in general, is an intractable problem. One modification of this approach is to subsample the constraints and then project onto the sampled set (see Nedić (2011); Polyak (2001); Wang & Bertsekas (2013); Wang et al. (2015)).

For metric constrained problems, however, we have many more constraints than data points, so the condition numbers of the problems are quite high and these algorithms tend to require a large number of iterations.

Another standard approach is to consider the Lagrangian and maintain a KKT type optimality condition. These methods run into two different kinds of problems. First, computing the gradient becomes an intractable problem for methods that maintain the KKT condition by using Newton's method. Examples of such methods include the interior point method and the barrier method. One fix could be to subsample the constraints and only compute those gradients, but this approach runs into the same drawbacks as before. The other option is to incrementally update the Lagrangian, looking at one constraint at a time. These methods, such as Bauschke & Lewis (2000); Iusem (1991); Iusem & De Pierro (1990), traditionally require us to cycle through all the constraints which is not feasible with metric constraints.

In this paper, we provide an algorithm, PROJECT AND FORGET, that uses Bregman projections with cutting planes, to solve metric constrained problems with many (possibly exponentially) inequality constraints. In fact, the algorithm is a general purpose one that can solve large constrained convex optimization problems, not only those arising from metric constraints. We also develop a stochastic version of our algorithm. This version is a similar adaptation of the Bregman method, as Nedić (2011); Polyak (2001); Wang & Bertsekas (2013); Wang et al. (2015) are adaptations of the projected gradient method. This version of our algorithm can be used to solve problems where each data point (or pair, triple of data points) form a constraint. The major contributions of our paper is as follows:

1. Using a specific instantiation of the PROJECT AND FORGET algorithm, we solve the weighted correlation clustering problem on a graph with over $130,000$ nodes. To solve this problem with previous methods, we would need to solve a linear program with over 10^{15} constraints. Furthermore, we demonstrate our algorithms superiority by outperforming the current state of the art in terms of CPU times.
2. We use our algorithm to develop a new algorithm that solves the metric nearness problem. We show that our algorithm outperforms the current state of the art with respect to CPU time and can be used to solve the problem for non-complete graphs.
3. We use the the stochastic version of our algorithm to develop a new algorithm to solve the information theoretic machine learning problem. We compare this against the standard method and show that in general we require fewer projections to solve the problem. Using our algorithm we can also solve the full version of the convex program presented in Davis et al. (2007) instead of a heuristic approximation. Thus, demonstrating that we can solve larger instances of the problem.
4. Finally, we prove that our algorithm converges to the global optimal solution. Additionally, we show that the optimality error (L_2 distance of the current iterate to the optimal) asymptotically decays at an exponential rate. We also show that because of the FORGET step, when the algorithm terminates, the set of constraints that remain remembered are exactly the active constraints. Thus, our algorithm also finds the set of active constraints.

We present the necessary background material and problem formulations in Section 2. In Section 3, we provide a general form of the PROJECT AND FORGET algorithm and detail its theoretical analysis. We instantiate our algorithm to solve three types of metric constrained problems in Section 4 and highlight the empirical performance. Complete proofs and discussion may be found in the sections in the Appendix.

2 PRELIMINARIES

2.1 METRIC CONSTRAINED PROBLEMS, GENERAL FORMULATION

Metric polytope. To set the stage for our optimization problems, we define the set over which we optimize first. Let $\text{MET}_n \subset \mathbb{R}^{\binom{n}{2}}$ be the space of all metrics on n points. Given a graph G the metric polytope $\text{MET}(G)$ is the projection of MET_n onto the coordinates given by the edges of G (i.e., we consider distances only between pairs of points that are adjacent in G).

It can be easily seen that for any $x \in \mathbb{R}^{\binom{n}{2}}$, we have that $x \in \text{MET}_n(G)$ if and only if $\forall\, e \in G$, $x(e) \geq 0$ and for every cycle C in G and $\forall\, e \in C$, we have that $x(e) \leq \sum_{\tilde{e} \in C, \tilde{e} \neq e} x(\tilde{e})$. Therefore, $\text{MET}_n(G)$ can be described as the intersection of exponentially many half-spaces.

Metric constrained problems. Now that we have the set over which we want to optimize, we give a general formulation for metric constrained optimization problems: given a strictly convex function f, a graph G, and a finite family of half-space $\mathcal{H} = \{H_i\}$ such that $H_i = \{x : \langle a_i, x\rangle \leq b_i\}$, we seek the unique point $x^* \in \bigcap_i H_i \cap \text{MET}(G) =: C$ that minimizes f. That is, if we set A to be the matrix whose rows are a_i and b be the vector whose coordinates are b_i we seek

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax \leq b \\ & x \in MET(G). \end{array} \tag{2.1}$$

The constraints encoded in the matrix A let us impose additional constraints beyond that of a metric. In this paper, A is used only for correlation clustering.

2.2 Specific metric constrained problems

Metric nearness. Following Brickell et al. (2008), the metric nearness problem is: given a point $x \in \mathbb{R}^{\binom{n}{2}}$, find the closest (in some ℓ_p norm) point $x^* \in \text{MET}_n$ to x. This problem is a form of metric learning; see Brickell et al. (2008) for an application to clustering and see Gilbert & Sonthalia (2018) for an application to unsupervised metric learning. I

Weighted correlation clustering on graphs. Bansal et al. (2004) introduced correlation clustering. In this problem, we are given a graph $G = (V, E)$ (not necessarily complete) in which each edge e has two non-negative numbers $w^+(e)$ and $w^-(e)$ that indicate the level of similarity and dissimilarity between its nodes. The goal of correlation clustering is to partition the nodes into clusters so as to minimize some objective function. The most common objective is $\displaystyle\sum_{e\in E} w^+(e)x_e + w^-(e)(1 - x_e)$, where $x_e \in \{0, 1\}$ indicates whether the end points of the edge e belong to different cluster. In general, this variant of the problem is NP-hard and many different algorithms have been developed to solve the problem. The best approximation results (with approximation ratios $O(\log n)$ in general, and $O(1)$ for specific cases), however, are obtained by rounding the solution to the following relaxed linear problem

$$\begin{array}{ll} \text{minimize} & \sum_{e\in E} w^+(e)x_e + w^-(e)(1 - x_e) \\ \text{subject to} & x_{ij} \leq x_{ik} + x_{kj} \qquad i, j, k = 1, ..., n \\ & x_{ij} \in [0, 1] \qquad i, j = 1, ..., n. \end{array} \tag{2.2}$$

See Charikar et al. (2005); Emanuel & Fiat (2003) for details. Many special cases, such as when the weights are ±1 and $G = K_n$, can be solved with faster algorithms (e.g., Ailon et al. (2005)).

Metric learning. The final metric constrained problem we consider is metric learning. There are many different versions of this problem (see Bellet et al. (2013); Suárez Díaz et al. (2018) for two different surveys on the topic) but all of the instantiations have a similar formulation: given a data set X and possibly some additional information, learn an appropriate metric on X. Many of the existing methods seek a linear map L such that the learned metric is given by

$$d_C(x, y) = \sqrt{(x - y)^T C(x - y)}$$

where $C = L^T L$. Our general purpose algorithm can directly learn the appropriate metric without resorting to the above specific form, however, this generalization requires more focus than we can give it in this paper. Instead, we focus on a specific instantiation (that also differs from the above), specifically information theoretic metric learning (ITML), as in Davis et al. (2007). In ITML, we consider $L^T L$ as the covariance matrix of a Gaussian distribution $p(x; C)$ and we are given two sets S, D which represent the set of similar and dissimilar points. The problem we solve is:

$$\begin{array}{ll} \text{minimize} & KL\big(p(x; C) \| p(x; I)\big) \\ \text{subject to} & d_A(x_i, x_j) \leq u \qquad (i, j) \in S \\ & d_A(x_i, x_j) \geq l. \qquad (i, j) \in D \end{array} \tag{2.3}$$

2.3 Bregman Projections

All of the above problems can be couched in general terms and, in the Appendix 6, we give such a general formulation. We seek to optimize a rich class of convex functions f, known as Bregman functions, denoted $\mathcal{B}(S)$, with useful properties for algorithmic and convergence analysis, subject to metric constraints. More details about Bregman functions and the general setting in which our algorithm works can be found in the Appendix sec:generalProblem . We do, however, detail Bregman projections, a key step in our specific algorithms in this section.

Definition 1. *Given a convex function $f(x) : S \to \mathbb{R}$ whose gradient is defined on all of S, we define its generalized Bregman distance $D_f : S \times S \to \mathbb{R}$ as follows: $D_f(x, y) = f(x) - f(y) - \langle \nabla f(y), x - y \rangle$.*

Definition 2. *Given a strictly convex function f, a closed convex set C, and a point y, the projection of y onto C with respect to D_f is a point $x^* \in \operatorname{dom}(f)$ such that*

$$x^* = \underset{x \in C \cap \operatorname{dom}(f)}{\arg\min} \; D_f(x, y).$$

3 Algorithms and analysis

In this section, we present our algorithm that we will use to solve metric constrained problems and state its convergence behavior. The detailed proofs can all be found in the Appendix 8.

3.1 The Algorithm

Our method is an iterative one and is presented in Algorithm 1. Each iteration consists of three phases. In the first phase, we obtain[1] a list of violated metric constraints L. If we have additional constraints represented in A, $\mathcal{A}$ is a list corresponding to these hyperplanes. In the second phase, we merge $L^{(\nu)}$, the list of constraints we have been keeping tracking of up to the νth iteration, with L and project onto each of the constraints in the list $L^{(\nu)} \cup L \cup \mathcal{A}$ iteratively. Finally, in the third phase, we forget some constraints.

Algorithm 1 General Algorithm.

```
function F($f$)
    $L^{(0)} = \emptyset$, $z^{(0)} = 0$. Initialize $x^{(0)}$ so that $\nabla f(x^{(0)}) = 0$.
    while Not Converged do
        $L = $ METRIC VIOLATIONS($x^{\nu}$)
        $\tilde{L}^{(\nu+1)} = L^{(\nu)} \cup L \cup \mathcal{A}$
        $x^{(\nu+1)} = $ Project($x^{(\nu)}, \tilde{L}^{(\nu+1)}$)
        $L^{(\nu+1)} = $ Forget($\tilde{L}^{(\nu+1)}$)
    return $x$

function METRIC VIOLATIONS($d$)
    $L = \emptyset$
    Let $d(i, j)$ be the weight of shortest path between nodes $i$ and $j$ or $\infty$ if none exists.
    for Edge $e = i(, j) \in E$ do
        if $w(i, j) > d(i, j)$ then
            Let $P$ be the shortest path between $i$ and $j$
            Add $C = P \cup \{(i, j)\}$ to $L$
    return $L$
```

The project and forget steps for algorithm are presented in Algorithm 2. Let us step through the code to understand intuitively its behavior. Let $H_i = \{x : \langle a_i, x \rangle \leq b_i$ be a constraint and x the current iterate. The first step is to calculate x^* and θ. Here x^* is the projection of x onto the boundary of H_i and θ is a "measure" of how far x is from x^*. However, θ can be any real number and so we examine two cases: θ positive or negative.

It can be easily seen that θ is negative if and only if the constraint is violated. In this case, we have $c = \theta$ because (as we will see in proof) the algorithm always maintains $z_i \geq 0$. Then on line 5, we compute the projection of x onto H_i. Finally, since we corrected x for this constraint, we add $|\theta|$ to z_i. Since each time we correct for H_i, we add to z_i, we see that z_i stores the total corrections made for H_i.

On the other hand, if θ is positive, this constraint is satisfied. In this case, if we also have that z_i is positive; i.e., we have corrected for H_i before, then we have over compensated for this constraint and, thus, we must undo some of the corrections. If $c = z_i$, then we undo all of the corrections and z_i is set to 0. Otherwise, if $c = \theta$ we only undo part of the correction.

[1] In the general formulation of the algorithm in the Appendix 6, this list is obtained by querying a separation oracle with one of two properties. See the Appendix for a detailed discussion.

The forget step is relatively easy: given a constraint H_i, we check if $z_i = 0$. If $z_i = 0$, then this means we have not done any net corrections for this constraint and we can forget about it; i.e., delete it from $L^{(\nu)}$.

Algorithm 2 Project and Forget algorithms.

```
1: function PROJECT($x, z, L$)
2:     for $H_i = \{y : \langle a_i, y\rangle = b_i\} \in L$ do
3:         Find $x^*, \theta$ by solving $\nabla f(x^*) - \nabla f(x) = \theta a_i$ and $x^* \in H_i$
4:         $c_i = \min(z_i, \theta)$
5:         $x \leftarrow$ such that $\nabla f(x^{n+1}) - \nabla f(x) = c_i a_i$
6:         $z_i \leftarrow z_i - c_i$
       return $x$, $z$
7: function FORGET($x, z, L$)
8:     for $H_i = \{x : \langle a_i, x\rangle = b_i\} \in L$ do
9:         if $z_i == 0$ then Forget $H_i$
       return $L$
```

In general, calculating the Bregman projection (line 3) cannot be done exactly. See Dhillon & Tropp (2007) for a general method to perform the calculation on line 3 and for an analytic formula for when f is a quadratic function.

3.2 Convergence Analysis

Now that we have specified the algorithm, we establish a few crucial theoretical properties. The first is that our algorithm is guaranteed to converge to the global optimum solution. In fact, we also show that asymptotically, our error decreases at an exponential rate. These main theoretical results can be summarized by the following theorem.

Theorem 1. *If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f*[2]*, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then*

1. *Then any sequence x^n produced by Algorithm 1 converges (with probability 1) to the optimal solution of problem 2.1.*
2. *If x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semidefinite, then there exists $\rho \in (0, 1)$ such that*

$$\lim_{\nu\to\infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \le \rho \tag{3.1}$$

where $\|y\|_H^2 = y^T H y$.

Proof. The proof of this theorem has been moved to the supplementary material section. □

The proof of Theorem 1 also establishes another important theoretical property.

Proposition 1. *If a_i is an inactive constraint, then (with probability 1) $z_i^{\nu} = 0$ for the tail of the sequence.*

Corollary 1. *Under the assumptions for part (2) of Theorem 1, we have that the sequence $z^n \to z^*$ also converges.*

These properties are important as they permit the following interpretation of our algorithm. The algorithm spends the initial few iterations identifying the active constraints from amongst a large number of constraints. The algorithm then spends the remainder of the iterations finding the optimal solution with respect to these constraints. This ability to find the set of active constraints is of the main advantages of our algorithm.

3.3 Stochastic Variant

In some problems, we do not optimize over the whole of MET(G) but a subset, as is the case in ITML. In such problems, we have constraints defined using subsets of the data points and as we may have many data points, we may have considerably more constraints than we want to examine. For this reason, we present a stochastic version of

[2] This is a technical condition and is discussed in the appendix refsec:genealProblem. In the case of our problems this is true

our algorithm that can be used in these cases. Instead of calling METRIC VIOLATION to get a list of metric violated constraints, we randomly sample from our constraints. In this version, at each iteration, we choose a random set of constraint and project onto these constraints and the ones we remember from before, and then forget some constraints. The advantage this has over similar stochastic methods such Wang et al. (2015) is that as we sample constraints, the list $L^{(\nu)}$ keeps track of the important constraints that we have seen so far. In this case, we have the following convergence result.

Theorem 2. *If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then with probability 1 any sequence x^n produced by the stochastic algorithm converges to the optimal solution of problem 2.1. Furthermore, if x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semidefinite, then there exists $\rho \in (0,1)$ such that with probability 1,*

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^\nu\|_H} \leq \rho.$$

4 EXPERIMENTS

To demonstrate the effectiveness of our method in solving metric constrained problems, we solve large instances of each of the problem. More details[3] about each of the experiments can be found in the Appendix 9.

4.1 WEIGHTED CORRELATION CLUSTERING ON GENERAL GRAPHS

4.1.1 DENSE GRAPHS

As the LP formulation for correlation clustering in Equation 2.2 has $O(n^3)$ constraints, solving the LP for large n becomes infeasible quickly, in terms of both memory and time. Veldt et al. (2019) showed that for instances with $n \approx 4000$, standard solvers such as Gurobi ran out of memory on a 100 GB machine. On the other hand, Veldt et al. (2019) develop a method using which they can feasibly solve the problem for $n \approx 11000$. To solve the problem, they transformation problem 2.2 into problem 4.1 for some an appropriately defined $d, \tilde{w}, W$. For general γ, the solution to problem 4.1 approximates the optimal solution to 2.2. However, for large enough γ it has been shown that the two problems are equivalent.

$$\begin{array}{ll} \text{minimize} & \tilde{w}^T|x-d| + \frac{1}{\gamma}|x-d|^T W |x-d| \\ \text{subject to} & x \in \text{MET}(K_n) \end{array} \tag{4.1}$$

This is the version of the LP that we solve with our algorithm. We solve this for four graphs from the Stanford sparse network repository Leskovec & Krevl (2014). Following Veldt et al. (2019), we use the method from Wang et al. (2013) to convert these graphs into instances of weighted correlation clustering on a complete graph. We compare our method against Ruggles et al. (2019), a parallel version of Veldt et al. (2019), in terms of running time, quality of the solutions, and memory usage.

We see from Table 1 that our algorithm takes less time to get a better approximation ratio, but requires more memory per iteration. Our algorithm requires more memory because the initial few iterations find a large number of constraints. Later, in the forget step, the algorithm forgets these constraints until the number of constraints stabilizes at a reasonable level. Hence, our initial memory usage is much larger than our later memory usage.

To see how the number of constraints found by the oracle evolves, we plot the number of constraints found by the oracle and the number of constraints after the forget step for the CA-HepTh graph. This plot can be seen in Figure 1. Figure 1 also shows us, as expected, *the exponential decay of the maximum violation of a metric constraint.*

4.1.2 SPARSE GRAPHS

For much real-world data, the graph G is larger than our previous experiments but it is also sparse. As stated, the problem (2.2) requires $O(n^3)$ constraints and $O(n^2)$ variables. However, in practice, there are very few active constraints, as seen in Table 2. Proposition 2 tells us that if we optimize over MET(G) instead of MET$_n$, then the quality of the solution is not degraded. With this generalization to MET(G), the number of variables is greatly reduced but the number

[3] All implementations and experiments can be found at `https://www.dropbox.com/sh/lq5nnhi4je2lh89/AABUUW7k5z3lXTSm8x1hhN1Da?dl=0`.

Table 1: Comparison with Ruggles et al. (2019). We set $\gamma = 1$ and ran until the maximum violation of a metric constraint was smaller than 0.01. We used a parallel version of the algorithm METRIC VIOLATION as our oracle. All of the computations were done on a machine with 16 physical cores and 13 GB of RAM per core. More details about the experiment can be found in the supplementary material section.

Graph		Time			Opt Ratio		Avg. mem. / iteration	
	n	Ours	Ruggles et al.	Speed up	Ours	Ruggles et al.	Ours	Ruggles et al.
CA-GrQc	4158	2098 s	5577 s	2.66	1.33	1.38	4.4 Gib	1.3 Gib
Power	4941	1393 s	6082 s	4.37	1.33	1.37	5.9 Gib	2 Gib
CA-HepTh	8638	9660 s	35021 s	3.62	1.33	1.36	24 Gib	8 Gib
CA-HepPh	11204	71071 s	135568 s	1.71	1.33	1.46	27.5 Gib	15 GIb

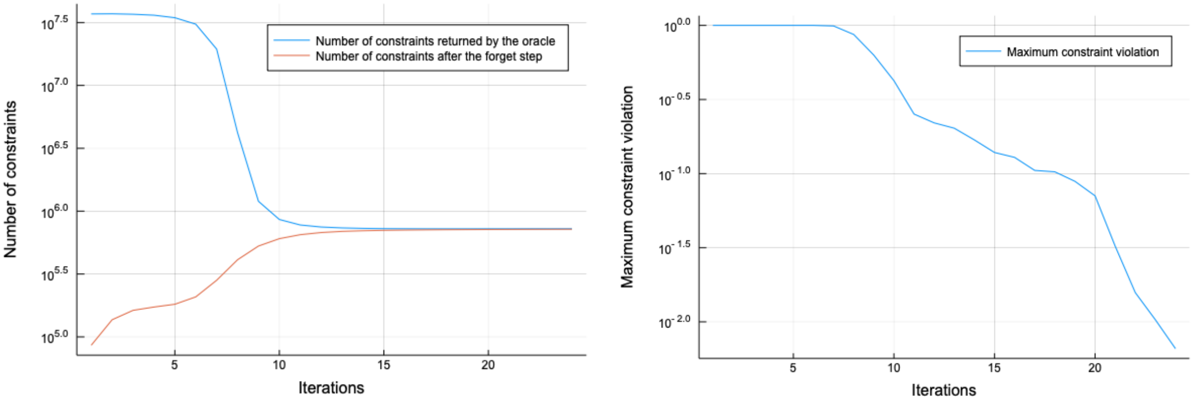

Figure 1: Left: The number of constraints is on the y axis (log scale) and the number of iterations on the x axis. Right: Plot for the maximum violation of a metric constraint (log scale) versus iterations.

of constraints is increased. Our algorithmic approach, however, can handle this increase in the number of constraints. Thus, we can solve the weighted correlation clustering problem for general graphs G.

Proposition 2. *Let π be the projection from MET_n to $MET(G)$. Then for any optimal solution x^* to the following problem*

$$\begin{array}{ll} \textit{minimize} & \sum_{e\in E} w^+(e)x_e + w^-(e)(1-x_e) \\ \textit{subject to} & x \in MET(G) \\ & x_{ij} \in [0,1], \qquad i,j = 1,...,n \end{array} \tag{4.2}$$

we have that for all $\hat{x} \in \pi^{-1}(x^*)$, $\hat{x}$ is an optimal solution to 2.2.

Table 2: Results for our algorithm on larger graphs. We ran our experiment on a machine with 48 physical cores and 13 GB of RAM per core, but we only used 32 threads for the computations.

Graph	n	# Constraints	Time	Opt Ratio	# Active Constraints	Iters.
Slashdot	82140	5.54×10^{14}	46.7 hours	1.78	384227	145
Epinions	131,828	2.29×10^{15}	121.2 hours	1.77	579926	193

Since the weighting of the edges does not affect the size of the linear program that needs to be solved, we tested our algorithm on signed graphs to get an estimate of the running time for the algorithm. We took two graphs from Leskovec & Krevl (2014). These graphs are much bigger instances than our previous experiments and have 82140 nodes and 131,828 nodes, respectively. Even if we use the parallel version of Veldt et al. (2019), based on the average time it took for a single iteration for the CA-HepPh graph, it would take an estimated two days for a single iteration for a graph with

$n \approx 80,000$. Since most graphs require at least 100 iterations, Veldt et al. (2019); Ruggles et al. (2019) cannot be used to solve problems of this magnitude. Other methods of solving the LP are also not feasible as they run out of memory on much smaller instances.

As we can see from Table 2 these instances have over 500 trillion constraints, but the number of active constraints is only a tiny fraction of the number of active constraints. Thus, using our approach we can solve the weighted correlation clustering problem on much larger graphs than ever before and this is possible because the graphs are sparse. That is, our oracle finds violated cycle inequalities relatively quickly and, since we forget inactive constraints, we project onto only a relatively small number of constraints. Thus, each iteration is done relatively quickly. In this experiment, each iteration took 500 to 3000 seconds.

4.2 Metric nearness

To do a head to head comparison against the algorithm presented in Brickell et al. (2008), we generated two types of random weighted complete graphs. For type one graphs, for each edge e we set $w(e) = 1$ with probability 0.8 and and $w(e) = 0$ with probability 0.2. For type two graphs, we let $w(e) \sim \mathcal{N}(0, 1)$. For both types of graphs, we ran both algorithms until the distance between the current iterate and its optimal decrease only solution (see Brickell et al. (2008)) was smaller than one. The computations were done on a machine with four physical cores with 13 GB of memory per core.

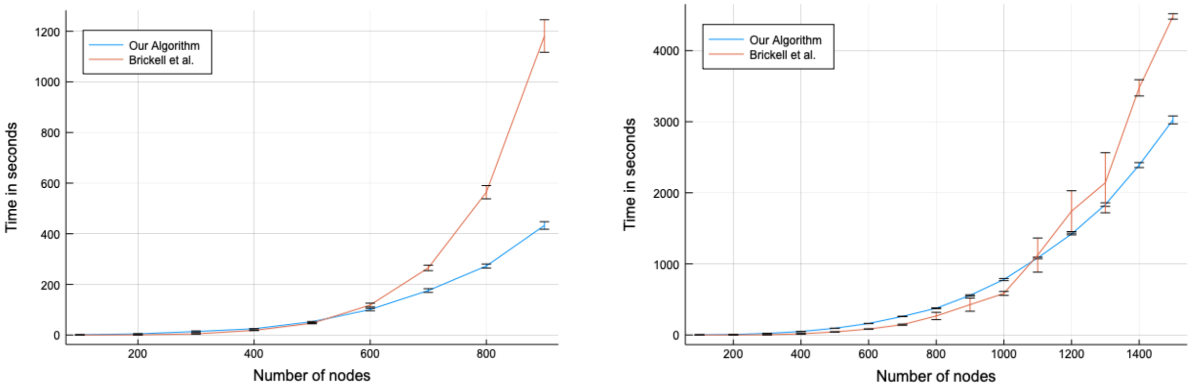

Figure 2: The red line is the mean running time over 5 samples for the algorithm from Brickell et al. (2008). The blue line is the running mean time for our algorithm. On the left we have the running times for type one graphs and on the right we have the running times for type two graphs.

We can see from Figure 2 that as n grows, our algorithm outperforms the algorithm from Brickell et al. (2008). We also see that our algorithm has less variability in its running time. Additionally, since we forget constraints, we are interested in the number of active constraints for these problems. From our experiments, we see that for type one graphs, our algorithm consistently returns $n^2/2$ constraints, and for type two graphs, consistently returns n^2 constraints.

In general, Brickell et al. (2008)'s algorithm works for $G = K_n$ only. For real-world data, we may not have full information about the interactions between data points; such relations may be sparse. Therefore, we do not restrict ourselves to finding the closest point in MET$_n$ when seeking a metric. Instead, we want to find the closest point in MET(G) where G is the graph representing the interactions for which we have information. As seen in Section 4.1.2 we can solve the problem when G is sparse.

4.3 Metric learning

For the final problem, we solve a variant of the metric learning problem, ITML. To do a head to head comparison against Davis et al. (2007), we implemented their algorithm in Julia. As the algorithm from Davis et al. (2007) doesn't solve the complete linear program, only an approximation, we compare our algorithms on the quality of the solution.

For each data set, we uniformly at random choose 80% of the data points to be the training set and the remaining to be the test set. We then let the similar pairs S be pairs that had the same label and the dissimilar pairs D be all of the other pairs. For the algorithm from Davis et al. (2007), as suggested in the paper, we randomly sampled $20c^2$ constraints, where c is the number of different classes and ran the algorithm so that it performed about 10^6 projections.

Because the algorithm from Davis et al. (2007) is also based on Bregman projections, to do a fair head to head comparison, we limit our selves to the same number of projections and, on each iteration, randomly sampled 2×10^5 constraints, 10^5 from S and 10^5 from D. We then ran the algorithm for 10 iterations. Note when sampling constraints we allowed repetition. That is, if a constraint was sampled k times during an iteration, we projected onto it k times.

Table 3: Table comparing the testing accuracy of PROJECT AND FORGET and ITML. Here the accuracy for ITML is the average over 100 trials.

Algorithm	Banana	Ionsphere	Coil2000	Letter	Penbased	Spambase	Texture
Project and Forget	**0.88679**	**0.90000**	**0.93842**	0.90250	0.97408	**0.93587**	**0.99909**
ITML	**0.88816**	0.86985	**0.93842**	**0.92852**	**0.99039**	0.90713	0.98390

The accuracy reported in Table 3 is based on one run of our algorithm since we have a theoretical guarantee that we converge to the optimal solution. However, since there is a great amount of variability in the constraints selected for the algorithm from Davis et al. (2007), we ran their algorithm 100 times and report the mean accuracy. As it can be seen from the table, in half cases our algorithm has better test accuracy as we solve the complete linear program.

These results are interesting as both methods are guaranteed theoretically to converge to their optimal solution; the difference in accuracy is due to the difference in the set of constraints being considered. We conclude that in the cases ITML does better on average, it indicates that subsampling the set of constraints, rather than using all of constraints, leads to lower generalization error. To verify that this difference in performance highlighted a difference in problems and not in convergence of the different algorithms, we ran PROJECT AND FORGET for 10^5 iterations, where we sample 2×10^3 constraints per iteration and did not see any significant improvement in the accuracy.

5 FUTURE WORK

In conclusion we see that our algorithm PROJECT AND FORGET can be used to solve not only large scale versions of three metric constrained problems but also opens many avenues for future work.

The first direction is motivated by ITML. As we see in Table 3, sometimes it is better to select only a small subset of the constraints rather than computing with all of them. It is an interesting question whether we can determine what inherent properties of the data set result in this phenomenon. If we can answer this, we can reduce the number of constraints by several orders of magnitude.

Another avenue of work that this algorithm opens is in learning better, more general metrics. Many existing metric learning algorithms learn a linear transform of Euclidean space and then use this transformed space to obtain a more relevant metric. Using our technique, however, we can directly learn a Euclidean metric that optimizes the objective function which we could then embed into Euclidean space using multidimensional scaling or some other embedding technique. We are also no longer restricted to Euclidean metrics, but can learn hyperbolic and tree metrics as well, getting better metric representations of our data. With these techniques, we would gain a better understanding the space a data set lies in and a better, usable metric representation.

REFERENCES

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pp. 684–693. ACM, New York, 2005. doi: 10.1145/1060590.1060692.

David Avis, Patrick Hayden, and Mark M. Wilde. Leggett-Garg inequalities and the geometry of the cut polytope. *Phys. Rev. A (3)*, 82(3):030102, 4, 2010. ISSN 1050-2947. doi: 10.1103/PhysRevA.82.030102.

Egon Balas and William Pulleyblank. The perfectly matchable subgraph polytope of a bipartite graph. In *Proceedings of the symposium on the matching problem: theory, algorithms, and applications (Gaithersburg, Md., 1981)*, volume 13, pp. 495–516, 1983. doi: 10.1002/net.3230130405.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004. ISSN 0885-6125. doi: 10.1023/B:MACH.0000033116.57574.95.

Francisco Barahona. On cuts and matchings in planar graphs. *Math. Programming*, 60(1, Ser. A):53–68, 1993. ISSN 0025-5610. doi: 10.1007/BF01580600.

Francisco Barahona and Ali Ridha Mahjoub. Compositions of graphs and polyhedra. I. Balanced induced subgraphs and acyclic subgraphs. *SIAM J. Discrete Math.*, 7(3):344–358, 1994. ISSN 0895-4801. doi: 10.1137/S0895480190182666.

Heinz H. Bauschke and Jonathan M. Borwein. Legendre functions and the method of random Bregman projections. *J. Convex Anal.*, 4(1):27–67, 1997. ISSN 0944-6532.

Heinz H. Bauschke and Adrian S. Lewis. Dykstra's algorithm with Bregman projections: a convergence proof. *Optimization*, 48(4):409–427, 2000. ISSN 0233-1934. doi: 10.1080/02331930008844513.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009. ISSN 1936-4954. doi: 10.1137/080716542.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *ArXiv*, abs/1306.6709, 2013.

Justin Brickell, Inderjit S. Dhillon, Suvrit Sra, and Joel A. Tropp. The metric nearness problem. *SIAM J. Matrix Anal. Appl.*, 30(1):375–396, 2008. ISSN 0895-4798. doi: 10.1137/060653391.

Yair Censor and Stavros Zenios. *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, 01 1997. ISBN ISBN-13: 978-0195100624.

Karthekeyan Chandrasekaran, László A. Végh, and Santosh Vempala. The cutting plane method is polynomial for perfect matchings. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science—FOCS 2012*, pp. 571–580. IEEE Computer Soc., Los Alamitos, CA, 2012.

Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *J. Comput. System Sci.*, 71(3):360–383, 2005. ISSN 0022-0000. doi: 10.1016/j.jcss.2004.10.012.

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 209–216, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

Santanu S. Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Math. Program.*, 170(1, Ser. B):237–266, 2018. ISSN 0025-5610. doi: 10.1007/s10107-018-1302-4.

Michel Deza and Monique Laurent. Applications of cut polyhedra. I, II. *J. Comput. Appl. Math.*, 55(2):191–216, 217–247, 1994. ISSN 0377-0427. doi: 10.1016/0377-0427(94)90020-5.

Inderjit S. Dhillon and Joel A. Tropp. Matrix nearness problems with Bregman divergences. *SIAM J. Matrix Anal. Appl.*, 29(4):1120–1146, 2007. ISSN 0895-4798. doi: 10.1137/060649021.

Tommy Elfving. An algorithm for maximum entropy image reconstruction from noisy data. *Math. Comput. Modelling*, 12(6):729–745, 1989. ISSN 0895-7177. doi: 10.1016/0895-7177(89)90358-0.

Dotan Emanuel and Amos Fiat. Correlation clustering—minimizing disagreements on arbitrary weighted graphs. In *Algorithms—ESA 2003*, volume 2832 of *Lecture Notes in Comput. Sci.*, pp. 208–220. Springer, Berlin, 2003. doi: 10.1007/978-3-540-39658-1_21.

Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, 6:1889–1918, 2005. ISSN 1532-4435.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *STOC'12—Proceedings of the 2012 ACM Symposium on Theory of Computing*, pp. 95–106. ACM, New York, 2012. doi: 10.1145/2213977.2213988.

A. C. Gilbert and L. Jain. If it ain't broke, don't fix it: Sparse metric repair. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 612–619, Oct 2017. doi: 10.1109/ALLERTON.2017.8262793.

Anna C. Gilbert and Rishi Sonthalia. Unsupervised metric learning in presence of missing data. *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 313–321, 2018.

W. Glunt, T. L. Hayden, S. Hong, and J. Wells. An alternating projection algorithm for computing the nearest Euclidean distance matrix. *SIAM J. Matrix Anal. Appl.*, 11(4):589–600, 1990a. ISSN 0895-4798. doi: 10.1137/0611042.

W. Glunt, T. L. Hayden, S. Hong, and J. Wells. An alternating projection algorithm for computing the nearest euclidean distance matrix. *SIAM Journal on Matrix Analysis and Applications*, 11:589–12, October 1990b.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. ISSN 0209-9683. doi: 10.1007/BF02579273.

Piotr Indyk. Sublinear time algorithms for metric space problems. In *Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999)*, pp. 428–432. ACM, New York, 1999. doi: 10.1145/301250.301366.

Alfredo N. Iusem. On dual convergence and the rate of primal convergence of Bregman's convex programming method. *SIAM J. Optim.*, 1(3):401–423, 1991. ISSN 1052-6234. doi: 10.1137/0801025.

Alfredo N. Iusem and Alvaro R. De Pierro. On the convergence properties of Hildreth's quadratic programming algorithm. *Math. Programming*, 47(1, (Ser. A)):37–51, 1990. ISSN 0025-5610. doi: 10.1007/BF01580851.

Ariel Jaimovich, Gal Elidan, Hanah Margalit, and Nir Friedman. Towards an integrated protein-protein interaction network: A relational markov network approach. *Journal of computational biology : a journal of computational molecular cell biology*, 13 2:145–64, 2006.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77:27–59, 2009.

Monique Laurent. A connection between positive semidefinite and Euclidean distance matrix completion problems. *Linear Algebra Appl.*, 273:9–22, 1998. ISSN 0024-3795. doi: 10.1016/S0024-3795(98)90126-4.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, June 2014.

Bo Lin, Anthea Monod, and Ruriko Yoshida. Tropical foundations for probability and statistics on phylogenetic tree space. *arXiv e-prints*, 2018.

Angelia Nedić. Random algorithms for convex minimization problems. *Math. Program.*, 129(2, Ser. B):225–253, 2011. ISSN 0025-5610. doi: 10.1007/s10107-011-0468-9.

Yu. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269(3):543–547, 1983. ISSN 0002-3264.

Xinghao Pan, Dimitris S. Papailiopoulos, Samet Oymak, Benjamin Recht, Kannan Ramchandran, and Michael I. Jordan. Parallel correlation clustering on big graphs. In *NIPS*, 2015.

Divya Pandove, Shivani Goel, and Rinkle Rani. Correlation clustering methodologies and their fundamental results. *Expert Systems*, 2018.

Laurent Poirrier and James Yu. On the depth of cutting planes. *arXiv e-prints*, art. arXiv:1903.05304, Mar 2019.

B. T. Polyak. Random algorithms for solving convex inequalities. In *Inherently parallel algorithms in feasibility and optimization and their applications (Haifa, 2000)*, volume 8 of *Stud. Comput. Math.*, pp. 409–422. North-Holland, Amsterdam, 2001. doi: 10.1016/S1570-579X(01)80024-0.

Thomas Rothvoss. The matching polytope has exponential extension complexity. In *STOC'14—Proceedings of the 2014 ACM Symposium on Theory of Computing*, pp. 263–272. ACM, New York, 2014.

Cameron Ruggles, Nate Veldt, and David F. Gleich. A Parallel Projection Method for Metric Constrained Optimization. *arXiv e-prints*, art. arXiv:1901.10084, Jan 2019.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. *Proceedings of the 35th International Conference on Machine Learning*, pp. 4460–4469, July 2018.

I. J. Schoenberg. Metric spaces and positive definite functions. *Trans. Amer. Math. Soc.*, 44(3):522–536, 1938. ISSN 0002-9947. doi: 10.2307/1989894.

David Sontag and Tommi S. Jaakkola. New outer bounds on the marginal polytope. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1393–1400. Curran Associates, Inc., 2008.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening lp relaxations for map using message passing. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, pp. 503–510, Arlington, Virginia, United States, 2008. AUAI Press. ISBN 0-9749039-4-9.

David Sontag, Do Kook Choe, and Yitao Li. Efficiently searching for frustrated cycles in map inference. In *Proceedings of the Twenty-Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pp. 795–804, Corvallis, Oregon, 2012. AUAI Press.

David A Sontag and Tommi S. Jaakkola. On iteratively constraining the marginal polytope for approximate inference and map. *Semantic Scholar*, 2007.

Juan Luis Suárez Díaz, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms and software. *arxiv eprint*, 12 2018.

Yichuan Tang. Deep learning using support vector machines. *ICML 2013 Challenges in Representation Learning*, abs/1306.0239, 2013.

Nate Veldt, David Gleich, Anthony Wirth, and James Saunderson. Metric-constrained optimization for graph clustering algorithms. *SIAM Journal on Mathematics of Data Science*, 1:333–355, 01 2019. doi: 10.1137/18M1217152.

Mengdi Wang and Dimitri P. Bertsekas. Incremental constraint projection-proximal methods for nonsmooth convex optimization. *http://www.mit.edu/ dimitrib/*, 2013.

Mengdi Wang, Yichen Chen, Jialin Liu, and Yuantao Gu. Random multi-constraint projection: Stochastic gradient methods for convex optimization with many constraints. *ArXiv*, abs/1511.03760, 2015.

Yubo Wang, Linli Xu, Yucheng Chen, and Hao Wang. A scalable approach for general correlation clustering. In *ADMA*, 2013.

Chen Yanover, Talya Meltzer, and Yair Weiss. Linear programming relaxations and belief propagation—an empirical study. *J. Mach. Learn. Res.*, 7:1887–1907, 2006. ISSN 1532-4435.

6 Convex programming

As we mentioned in main body of the paper, there is a much more general formulation of the metric constrained optimization problems and a correspondingly general variant of the algorithm. We present that setting below and detail the convergence results in the following section.

Given a strictly convex function f, and a finite family of half-space $\mathcal{H} = \{H_i\}$ such that $H_i = \{x : \langle a_i, x\rangle \le b_i\}$, we want to find the unique point $x^* \in \bigcap_i H_i =: C$ that minimizes f. That is, if we set A to be the matrix whose rows are a_i and b be the vector whose coordinates are b_i we seek

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax \le b \end{array} \tag{6.1}$$

We refer to each H_i as a *constraint set* and C as the *feasible region*. We shall assume that C is not empty, i.e., there is at least one feasible point. In general, we can represent any half-space by a linear inequality.

In general, since A is an extremely large matrix, computing A or writing A down is not computationally feasible for many large instances of the problem. Therefore, we access the constraint sets only through an oracle that has one of the two following separation properties.

Property 1. *$\mathcal{Q}$ is a deterministic separation oracle for a family of half-spaces $\mathcal{H}$, if there exists a positive, non-decreasing, continuous function ϕ, with $\phi(0) = 0$, such that on input $x \in \mathbb{R}^d$, $\mathcal{Q}$ either certifies $x \in C$ or returns a list $\mathcal{L} \subset \mathcal{H}$ such that*

$$\max_{\tilde{C} \in \mathcal{L}} \operatorname{dist}(x, \tilde{C}) \ge \phi(\operatorname{dist}(x, C)).$$

Property 2. *$\mathcal{Q}$ is a random separation oracle for a family of half-spaces $\mathcal{H}$, if there exists some distribution $\mathcal{D}$ and a lower bound $\tau > 0$, such that on input $x \in \mathbb{R}^d$, $\mathcal{Q}$ returns a list $\mathcal{L} \subset \mathcal{H}$ such that*

$$\forall \tilde{H} \in \mathcal{H},\ Pr_{\mathcal{D}}[\tilde{H} \in \mathcal{L}] \ge \tau.$$

For the random oracle, we do not need it to decide whether $x \in C$, so this oralce can be used to solve optimization problems over polytopes for which deciding membership is NP-hard.

In addition our method works for a rich class of functions known as Bregman functions.

Definition 3. *A function $f : \Lambda \to \mathbb{R}$ is called a Bregman function if there exists a non-empty convex set S such that $\overline{S} \subset \Lambda$ and the following hold:*

(i) *$f(x)$ is continuous, strictly convex on $\overline{S}$, and has continuous partial derivatives in S.*

(ii) *For every $\alpha \in \mathbb{R}$, the partial level sets $L_1^f(y, \alpha) := \{x \in \overline{S} : D_f(x, y) \le \alpha\}$ and $L_2^f(x, \alpha) := \{y \in S : D_f(x, y) \le \alpha\}$ are bounded for all $x \in \overline{S}, y \in S$.*

(iii) *If $y_n \in S$ and $\lim_{n\to\infty} y_n = y^*$, then $\lim_{n\to\infty} D_f(y^*, y_n) = 0$.*

(iv) *If $y_n \in S$, $x_n \in \overline{S}$, $\lim_{n\to\infty} D_f(x_n, y_n) = 0$, $y_n \to y^*$, and x_n is bounded, then $x_n \to y^*$.*

We denote the family of Bregman functions by $\mathcal{B}(S)$. We refer to S as the zone of the function and we take the closure of the S to be the domain of f.

Definition 4. *We say that a hyperplane H_i is* strongly zone consistent *with a respect to a Bregman function f and its zone S, if for all $y \in S$ and for all hyperplanes H, parallel to H_i that lie in between y and H_i, the Bregman projection of y onto H lies in S instead of in $\overline{S}$.*

Next, we briefly discuss under what assumptions can we solve solve our optimization problem and what these assumptions mean.

Assumption 1. *$f(x)$ is a Bregman function.*

This class of function includes many natural objective functions, including $f(x) = -\sum_{i=1}^n x_i \log(x_i)$ with zone $S = \mathbb{R}^n_+$ (here f is defined on the boundary of S by taking the limit) and $f(x) = \frac{1}{p}\|x\|_p^p$ for $p \in (1, \infty)$. The ℓ_p norms for $p = 1, \infty$ are not Bregman functions but can be made Bregman functions by adding a quadratic term. That is, $f(x) = c^T x$ is a not Bregman function, but $c^T x + x^T Q x$ for any positive definite Q is a Bregman function.

Additionally, strongly convex functions and Legendre functions are related to Bregman functions, but neither class implies the other. See Bauschke & Lewis (2000) for an example and a more in-depth discussion.

Assumption 2. *All hyperplanes in $\mathcal{H}$ are strongly zone consistent with respect to $f(x)$.*

This assumption is used to guarantee that when we do a projection the point we project onto lies within our domain. This is not too restrictive. For example, all hyperplanes are strongly zone consistent with respect to the objective functions $f(x) = 0.5\|x\|^2$ and $f(x) = -\sum_i x_i \log(x_i)$.

Assumption 3. *C is non-empty.*

This is needed to make sure that the algorithm converges.

7 The general algorithms

To set the stage for subsequent discussions, we present the general structure of all of our algorithms. They are all iterative and, in general, are run until some convergence criterion has been met. The convergence criterion depends largely on the specific application for which the algorithm is tailored. For this reason, we postpone the discussion of the convergence criterion until the applications section.

Each iteration consists of three phases, as shown in algorithm 3. In the first phase, we query our oracle $\mathcal{Q}$ to obtain a list of constraints L. In the second phase, we merge $L^{(\nu)}$, the list of constraints we have been keeping tracking of up to the νth iteration, with L and project onto each of the constraints in the list $L^{(\nu)} \cup L$ iteratively. Finally, in the third phase, we forget some constraints.

Algorithm 3 General Algorithm.

```
function F($\mathcal{Q}, f$)
    $L^{(0)} = \emptyset$, $z^{(0)} = 0$. Initialize $x^{(0)}$ so that $\nabla f(x^{(0)}) = 0$.
    while Not Converged do
        $L = \mathcal{Q}(x^\nu)$
        $\tilde{L}^{(\nu+1)} = L^{(\nu)} \cup L$
        $x^{(\nu+1)}$ = Project($x^{(\nu)}, \tilde{L}^{(\nu+1)}$)
        $L^{(\nu+1)}$ = Forget($\tilde{L}^{(\nu+1)}$)
    return $x$
```

7.1 Project and Forget algorithms

The project and forget steps for algorithm are presented in Algorithm 4. Let us step through the code to understand the behavior of the algorithm. Let $H_i = \{x : \langle a_i, x\rangle \leq b_i$ be our constraint and x our current iterate. The first thing we do is calculate x^* and θ. Here x^* is the projection of x onto the boundary of H_i and θ is a "measure" of how far x is from x^*. However, θ can be any real number. Let us discuss separately the cases θ positive and negative below.

It can be easily seen Censor & Zenios (1997) that θ is negative if and only if the constraint is violated. In this case, $c = \theta$. This is because (as we will see in proof) we will always maintain that $z_i \geq 0$. Then on line 5, we just compute the projection of x onto H_i. Finally, since we corrected x for this constraint, we add $|\theta|$ to z_i. Since each time we correct for H_i, we add to z_i, we see that z_i stores how the total corrections made for H_i.

On the other hand, if θ is positive, this means that this constraint is satisfied. In this case, if we also have that z_i is positive, i.e., we have corrected for H_i before, then we have over compensated for this constraint. Hence we must do undo some of the corrections. Thus, if $c = z_i$ then we undo all of the corrections and z_i is now 0. Otherwise if $c = \theta$ we only undo part of the correction.

The FORGET step is relatively easy, given a constraint H_i we just need to check if $z_i = 0$. If $z_i = 0$, then this means we have not done any net corrections for this constraint. Thus, we can forget about it; i.e., delete it from $L^{(\nu)}$.

In general, calculating the Bregman projection (line 3) cannot be done exactly. See Dhillon & Tropp (2007) for a general method to perform the calculation on line 3 and for an analytic formula for when f is a quadratic function.

Algorithm 4 Project and Forget algorithms.

```
function PROJECT($x, z, L$)
    for $H_i = \{y : \langle a_i, y\rangle = b_i\} \in L$ do
        Find $x^*, \theta$ by solving $\nabla f(x^*) - \nabla f(x) = \theta a_i$ and $x^* \in H_i$
        $c_i = \min(z_i, \theta)$
        $x \leftarrow$ such that $\nabla f(x^{n+1}) - \nabla f(x) = c_i a_i$
        $z_i \leftarrow z_i - c_i$
    return $x, z$
function FORGET($x, z, L$)
    for $H_i = \{x : \langle a_i, x\rangle = b_i\} \in L$ do
        if $z_i == 0$ then Forget $H_i$
    return $L$
```

7.2 Convergence Analysis

Now that we have specified our algorithm, we need to establish a few crucial theoretical properties. The first is that under assumptions 1,2,3 our algorithm is guaranteed to converge to the global optimum solution. In fact, we also show that asymptotically, our error decreases at an exponential rate. These main theoretical results can be summarized by the following theorem.

Theorem 3. *If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then*

1. *If the oracle $\mathcal{Q}$ satisfies property 1 (property 2), then any sequence x^n produced by the above algorithm converges (with probability 1) to the optimal solution of problem 6.1.*
2. *If x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semidefinite, then there exists $\rho \in (0, 1)$ such that*

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho \tag{7.1}$$

where $\|y\|_H^2 = y^T H y$. In the case when we have an oracle that satisfies property 2, the limit in 7.1 holds with probability 1.

Proof. The proof of this theorem has been moved to the supplementary material section. □

The proof of Theorem 3 also establishes another important theoretical property.

Proposition 3. *If a_i is an inactive constraint, then (with probability 1) $z_i^\nu = 0$ for the tail of the sequence.*

Corollary 2. *Under the assumptions for part (2) of Theorem 3, we have that the sequence $z^n \to z^*$ also converges.*

These properties are important as they permit the following interpretation of our algorithm. The algorithm spends the initial few iterations identifying the active constraints from amongst a large number of constraints. The algorithm then spends the remainder of the iterations finding the optimal solution with respect to these constraints. This ability to find the set of active constraints is one of the main advantages of our algorithm.

7.3 The Metric Constrained Versions

First we can easily see that function METRIC VIOLATION in Algorithm 1 is an oracle that satisfies Property 1

Proposition 4. *Function* METRIC VIOLATION *in Algorithm 1 is an $\Theta(n^2 \log(n) + n|E|)$ oracle that has Property 1*

Proof. The first step in METRIC VIOLATION is to calculate the shortest distance between all pairs of nodes. This can be done using Dijkstra's algorithm in $\Theta(n^2 \log(n) + n|E|)$ time. Then if the shortest path between any adjacent pair of vertices is not the edge connecting them, then the algorithm has found a violated cycle inequality. Note that if no such path exists, then all cycle inequalities have been satisfied. Hence our point is within the metric polytope. Thus, we have an oracle that separates the polytope. However, we want an oracle that satisfies property 1.

Given a point x and a hyperplane $H_{C,e}$, defined by some cycle C and an edge e, the deficit of this constraint is given by the following formula.

$$d(C,e) = x(e) - \sum_{\tilde{e}\in C, \tilde{e}\neq e} x(\tilde{e})$$

If this quantity is positive, then x violates this constraint. In this case, the distance from x to this constraint is $\frac{d(C,e)}{|C|}$. While the above oracle does not find the cycle C and edge e that maximize this distance, it does find the cycle C and edge e that maximize $d(C,e)$. Since $|C| \in [1,n]$, if we let $\phi(x) = x/n$, we see that oracle satisfies property 1. □

Similarly we can see that uniformly randomly sampling constraints is an oracle that satisfies Property 2. Thus we can see that both algorithms presented in the main text are special cases of the above algorithm.

7.4 Truly Stochastic Variant

Now Algorithm 4 can be used with an oracle that satisfies property 2. However, this is not a completely stochastic algorithm. We still have to keep track of the constraints that we have seen and carefully pick which constraints to forget. Nevertheless, we can modify our forget step to forget all constraints and obtain a truly stochastic version of the algorithm. In this version, at each iteration, we choose a random set of constraint and project onto these constraints only, independently of what constraints were used in previous iterations. We cannot, however, forget the values of the dual variables. In this case, we have the following convergence result.

Theorem 4. *If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then with probability 1 any sequence x^n produced by the above truly stochastic algorithm converges to the optimal solution of problem 6.1. Furthermore, if x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semidefinite, then there exists $\rho \in (0,1)$ such that with probability 1,*

$$\liminf_{\nu\to\infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^{\nu}\|_H} \leq \rho.$$

This version of our algorithm is very similar to the algorithms presented Nedić (2011); Wang et al. (2015). The major difference being that we do not need to a gradient descent step. Instead we maintain the KKT conditions, by keeping track of the dual variables and doing dual corrections.

However, in practice using 4 with the random oracle tends to produce better results. This is because by not forgetting the active constraints that we have seen, instead of hoping that we sample them, we speed up convergence significantly.

8 Convergence Results

In this section, we present the proofs of Theorems 3 and 4. Because the proof of Theorem 3 is quite technical and involves two different types of separation oracles, we split it into several parts. In Subsections 8.1 and 8.2, we prove the first part of Theorem 3 for separation oracles with property 1 and 2, respectively. In Subsection 8.3, we prove the second part of Theorem 3 (also subdividing this proof into several cases). Finally, in Subsection 8.4 we prove Theorem 4, noting only the changes necessary from the proof of Theorem 3.

8.1 Proof of part 1 of Theorem 3 for oracles that satisfy property 1

We remind the reader of the notation established in Section 2. The vector of variables over which we optimize is x, f is the objective function, $H_i = \{y : \langle y, a_i\rangle = b_i\}$ are the hyperplanes that lie on the boundaries of the half-space constraints, L is the Lagrangian, z is the dual variable, A is the matrix with rows given by a_i, and b is the vector with rows b_i.

Next, we clarify the indexing of the variables. Algorithm 3 has three steps per iteration and during the Project step there are multiple projections. When we want to refer to a variable after the νth iteration, it will have a superscript with a ν. When we refer to a variable after the $n(i,k)$th projection, we use the superscript $n(i,k)$. Finally, before the nth projection, $i(n)$ will represent the index of the hyperplane onto which we project.

Finally, let R be the maximum number of constraints that our oracle $\mathcal{Q}$ returns. This is clearly upper bounded by the total number of constraints, which we have assumed is finite. We are now ready to prove the first part of Theorem 3.

Theorem 3. *(Part 1) If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, $\exists\, x^0 \in S$, such that $\nabla f(x^0) = 0$, and the oracle $\mathcal{Q}$ satisfies property 1, then any sequence x^n produced by Algorithm 3 converges to optimal solution of problem 6.1.*

Proof. The proof of this theorem is an adaptation of the proof of convergence for the traditional Bregman method that is presented in Censor & Zenios (1997) whose proof entails the following four steps. The main difference between Censor and Zenios' proof and ours is that of the last two steps. We present the entire proof, however, for completeness. To that end, we show that

Step 1. the KKT condition, $\nabla f(x) = \nabla f(x^0) - A^T z$, is always maintained,

Step 2. the sequence x^n is bounded and it has at least one accumulation point,

Step 3. any accumulation point of x^n is feasible (i.e., is in C), and

Step 4. any accumulation point is the optimal solution.

Step 1. The KKT condition, $\nabla f(x) = \nabla f(x^0) - A^T z$, is always maintained.

We show by induction that for all n, $\nabla f(x^n) = -A^T z^n$. In the base case, $z = 0$, thus, $\nabla f(x^0) = 0 = -A^T z^0$. Assume the result holds for iteration n, then

$$\nabla f(x^{n+1}) = \nabla f(x^n) + c^n a_{i(n)} = -A^T z^n + A^T c^n e_{i(n)} = -A^T (z^n - c^n e_{i(n)}) = -A^T z^{n+1}$$

We know that $c^n \leq z^n_{i(n)}$; therefore, we maintain $z^{n+1} \geq 0$ as well.

Step 2. The sequence x^n is bounded and has an accumulation point.

To show that x^n is a bounded, we first show that $\big(L(x^n, z^n)\big)_n$ is a monotonically increasing sequence bounded from above. This observation results from the following string of equalities:

$$\begin{aligned} L(x^{n+1}, z^{n+1}) - L(x^n, z^n) &= f(x^{n+1}) - f(x^n) + \langle z^{n+1}, Ax^{n+1} - b\rangle - \langle z^n, Ax^n - b\rangle \\ &= f(x^{n+1}) - f(x^n) + \langle A^T z^{n+1}, x^{n+1}\rangle - \langle A^T z^n, x^n\rangle - \langle z^{n+1} - z^n, b\rangle \\ &= f(x^{n+1}) - f(x^n) - \langle \nabla f(x^{n+1}), x^{n+1}\rangle + \langle \nabla f(x^n), x^n\rangle + \langle c^n e_{i(n)}, b\rangle \\ &= f(x^{n+1}) - f(x^n) - \langle \nabla f(x^n) + c^n a_{i(n)}, x^{n+1}\rangle + \langle \nabla f(x^n), x^n\rangle + c^n b_{i(n)} \\ &= f(x^{n+1}) - f(x^n) - \langle \nabla f(x^n), x^{n+1} - x^n\rangle - \langle c^n a_{i(n)}, x^{n+1}\rangle + c^n b_{i(n)} \\ &= \underbrace{D_f(x^{n+1}, x^n)}_{(1)} + \underbrace{c^n (b_{i(n)} - \langle a_{i(n)}, x^{n+1}\rangle)}_{(2)} \end{aligned}$$

Next, we show that both terms (1) and (2) are non-negative. We know that D_f is always non-negative so we only need to consider term (2). There are two cases: (i) if $c^n = \theta^n$, then $x^{n+1} \in H_{i(n)}$ and $b_{i(n)} - \langle a_{i(n)}, x^{n+1}\rangle = 0$. On the other hand, (ii) if $c^n = z^n_{i(n)}$, then $b_{i(n)} - \langle a_{i(n)}, x^{n+1}\rangle \geq 0$ and $c^n \geq 0$. We can conclude that the difference between successive terms of $L(x^n, z^n)$ is always non-negative and, hence, it is an increasing sequence.

To bound the sequence, let y be a feasible point (i.e., $Ay \leq b$). (Note that this is the only place we use the assumption that the feasible set is not empty.) Then

$$\begin{aligned} D_f(y, x^n) &= f(y) - f(x^n) - \langle \nabla f(x^n), y - x^n\rangle \\ &= f(y) - f(x^n) + \langle z^n, Ay - Ax^n\rangle \\ &\leq f(y) - f(x^n) + \langle z^n, b - Ax^n\rangle. \end{aligned}$$

Rearranging terms in the inequality, we obtain a bound on the sequence $L(x^n z^n)$ from above:

$$L(x^n, z^n) = f(x^n) + \langle z^n, Ax^n - b\rangle \leq f(y) - D_f(y, x^n) \leq f(y).$$

Since the sequence $(L(x^n, z^n))_{n\in\mathbb{N}}$ is increasing and bounded, it is a convergent sequence and the difference between successive terms of the sequence goes to 0. Therefore,

$$\lim_{n\to\infty} D_f(x^{n+1}, x^n) = 0.$$

From the previous inequality we also have that

$$D_f(y, x^n) \leq f(y) - L(x^n, z^n) \leq f(y) - L(x^0, z^0) =: \alpha.$$

Using part (ii) of the definition of a Bregman function, we see that $L_2^f(y, \alpha)$ is bounded and since $(x^n)_{n\in\mathbb{N}} \in L_2^f(y, \alpha)$, x^n is a bounded sequence with an accumulation point.

Step 3. Any accumulation point x^* of x^n is feasible (i.e., is in C).

This is the only step in which we use the fact that our oracle satisfies property 1. Let x^* be some accumulation point for x^n and assume for the sake of contradiction that $Ax^* \not\leq b$. Let $\tilde{A}, \tilde{b}$ be the maximal set of constraints that x^* does satisfy; i.e.,

$$\tilde{A}x^* \leq \tilde{b}.$$

Let (x^{n_k}) be a subsequence such that $x^{n_k} \to x^*$ and H be a constraint that x^* violates. Define ϵ as

$$\epsilon := \phi(d(x^*, H)) > 0. \tag{8.1}$$

Because x^n is bounded, x^{n_k} is a convergent subsequence $x^{n_k} \to x^*$, and $D_f(x^{n+1}, x^n) \to 0$, by equation 6.48 from Censor & Zenios (1997) we see that for any t,

$$x^{n_k+t} \to x^*.$$

In particular, the proposition holds for all $t \leq 2|\tilde{A}| + 2 =: T$.

Let us consider an augmented subsequence $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+T}$, i.e., add in extra terms. Note that if $n_{k+1} - n_k \to \infty$, then this augmented sequence is not the entire sequence. We want to show that infinitely many of the terms in our augmented sequence satisfy a constraint *not* in $\tilde{A}$. Should this hold, then because we have only finitely many constraints, there exists at least one single constraint $\tilde{a}$ that is *not* in $\tilde{A}$, such that infinitely many terms of the augmented sequence all satisfy the single constraint $\tilde{a}$. Finally, because our augmented sequence converges to x^* and we are only looking at closed constraints, we must have that x^* also satisfies the constraint $\tilde{a}$. Thus, we would arrive at a contradiction of the maximality of $\tilde{A}$ and x^* would have to be in the feasible region.

To see that infinitely many of the terms in our augmented sequence satisfy a constraint *not* in $\tilde{A}$, let ν_k be the iteration in which the n_kth projection takes place. Note that we can assume without loss of generality that in any iteration, we project onto any constraint at most once. If this were not the case and we projected onto constraints more often, we would simply change the value of T to reflect this larger number of projections. Therefore, we have two possibilities for which iteration the $n_k + |\tilde{A}| + 1$st projection takes place and we consider each case below.

Case 1: The $n_k + |\tilde{A}| + 1$st projection, infinitely often, takes places in ν_kth iteration. Since we project onto each constraint at most once, one of the projections between the n_k and $n_k + |\tilde{A}| + 1$st projection must be onto a hyperplane defined by a constraint *not* represented in $\tilde{A}$ and amongst the terms $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+|\tilde{A}|+1}$, we must have a term that satisfies a constraint *not* in $\tilde{A}$ infinitely often.

Case 2: The $n_k + |\tilde{A}| + 1$st projection, infinitely often, takes place in $\nu_k + 1$st iteration or later.

If this projection happens in $\nu_k + 1$st iteration, consider the iteration in which we do the $n_k + T$th projection. If this projection also takes place in the $\nu_k + 1$st iteration, then we have done at least $|\tilde{A}| + 1$ projections in the $\nu_k + 1$st iteration. Hence, amongst $x^{n_k+|\tilde{A}|+1}, \ldots, x^{n_k+T}$, we must have a term that satisfies a constraint *not* in $\tilde{A}$.

If the $n_k + |\tilde{A}| + 1$st or the $n_k + T$th projection happens in the $\nu_k + 2$nd iteration or later, then between the n_kth and the $n_k + T$th projection, we must have projected onto all constraints returned by oracle in the $\nu_k + 1$st iteration. Therefore, we must have projected onto some hyperplane defined by $\hat{a}$ (for some constraint $\hat{C}$) such that

$$\hat{d}^{n_k} := d(x^{\nu_k+1}, \hat{C}) \geq \phi(d(x^{\nu_k+1}, C)).$$

Then there exists a sufficiently small $\delta > 0$, depending on $\tilde{A}, \tilde{b}, x^*$, such that if $\|y - x^*\| \leq \delta$, then

$$\tilde{A}y \leq \tilde{b} + \frac{\epsilon}{2}\mathbb{1},$$

where $\mathbb{1}$ is vector of all ones.

Since our augmented sequence converges to x^*, we know that there exists a K, such that for all $k \geq K$ and $t \leq T$, $\|x^{n_k+t} - x^*\| < \delta$. That is, for all $k \geq K$ and $t \leq T$,

$$\tilde{A}x^{n_k+t} \leq \tilde{b} + \frac{\epsilon}{2}\mathbb{1}. \tag{8.2}$$

Note x^{ν_k+1} is within our augmented sequence so if $\hat{a}$ is infinitely often in $\tilde{A}$, by equation 8.2, we have that infinitely often

$$\frac{\epsilon}{2} \geq \hat{d}^{n_k}.$$

Finally, because the augmented sequence converges to x^*,

$$\epsilon = \phi(d(x^*, H)) \leq \phi(d(x^*, C)) = \lim_{k\to\infty} \phi(d(x^{\nu_k+1}, C)) \leq \lim_{k\to\infty} \hat{d}^{n_k} \leq \frac{\epsilon}{2}.$$

The first inequality follows from the fact that ϕ is non-decreasing. Therefore, $\hat{a}$ is *not* in $\tilde{A}$ infinitely often and amongst $x^{n_k}, x^{n_k+1}, \ldots, x^{n_k+T}$, we must have a term that satisfies a constraint *not* in $\tilde{A}$ infinitely often. Thus, there is a constraint $\tilde{a}$ not in $\tilde{A}$ that is satisfied by infinitely terms of our augmented sequence and we have a contradiction.

Step 4. Optimality of accumulation point.

Because we have established the feasibility of all accumulation points, we show next that any accumulation point $x^{n_k} \to x^*$ is optimal.

First, we show that there exists an N, such that for any $k \geq N$, and for any a_i such that

$$\langle a_i, x^* \rangle < b_i,$$

we have $z_i^{n_k} = 0$. To do so, we assume for the sake of contradiction that for some a_i, our sequence $z_i^{n_k}$ is infinitely often not 0. The algorithm then projects onto this constraint infinitely often. Therefore, the point x^{n_k} lies on the hyperplane defined by a_i, b_i infinitely often. Thus, the limit point x^* must lie on this hyperplane as well and we have a contradiction.

Now we know that for any constraint a_i, we either have that $\langle a_i, x^* \rangle = b_i$ or we have that $z_i^{n_k} = 0$ for the tail of the sequence. Thus, for sufficiently large k,

$$\langle z^{n_k}, Ax^{n_k} - b \rangle = \langle A^T z^{n_k}, x^{n_k} - x^* \rangle = \langle -\nabla f(x^{n_k}), x^{n_k} - x^* \rangle = D_f(x^*, x^{n_k}) - f(x^*) + f(x^{n_k}).$$

Next, by part (iii) of the definition of a Bregman function,

$$\lim_{k\to\infty} D_f(x^*, x^{n_k}) = 0.$$

Finally,

$$\lim_{k\to\infty} L(x^k, z^k) = \lim_{k\to\infty} f(x^{n_k}) + \langle z^{n_k}, Ax^{n_k} - b \rangle = f(x^*).$$

We also know that $L(x^k, z^k) \leq f(y)$ for any feasible y. Thus, $f(x^*) \leq f(y)$. Hence x^* is an optimal solution. Now since f is strictly convex, this optimal point is unique. Therefore, we have that $(x^n)_{n\in\mathbb{N}}$ has only one accumulation point and $x^n \to x^*$. □

An important fact consequence of this proof is the following proposition:

Proposition 3. *If a_i is an inactive constraint, then there exists a N, such that for all $n \geq N$, we have that $z_i^n = 0$. That is, after some finite time, we never project onto inactive constraints ever again.*

8.2 Proof of part 1 of Theorem 3 for oracles that satisfy property 2

In this subsection, we prove part 1 of Theorem 3 for oracles that satisfy property 2. We make note of the key ideas in this proof as they are useful in the proof of the truly stochastic variant. To be precise, we prove:

Theorem 1. *(Part 1) If $f \in \mathcal{B}(S)$, H_i are strongly zone consistent with respect to f, $\exists\, x^0 \in S$, such that $\nabla f(x^0) = 0$, and the oracle $\mathcal{Q}$ satisfies property 2, then with probability 1, any sequence x^n produced by Algorithm 3 converges to optimal solution of problem 6.1.*

Proof. Assume that we have an oracle that satisfies property 2. A careful reading of the previous proof shows that if we switch out an oracle with property 1 for an oracle with property 2, then we only need to adjust step 3 of our proof. The crucial part of that step was showing that for our augmented sequence, we had infinitely many terms that satisfied a constraint *not* in $\tilde{A}$. We make the following adjustments to our analysis.

Let ν_k be the iteration in which the n_kth projection takes place. In the previous proof, we used the property of the oracle only when the $n_k + T$th projection took place in the $\nu_k + 2$nd iteration or a later iteration. In this case, the augmented sequence encompasses all of the $\nu_k + 1$st iteration infinitely often.

Let us choose a constraint $\hat{a}$ that is not satisfied by x^*. Because the oracle satisfies property 2, for each iteration $\nu_k + 1$, our oracle returns $\hat{a}$ with probability at least $\tau > 0$. By the Borel Cantelli Lemma, we know that during the selected iterations, the constraint $\hat{a}$ is, with probability one, returned infinitely often by our oracle. Thus, our augmented sequence satisfies this constraint with probability 1 and x^* lies in the feasible region with probability 1. □

A direct consequence of this proof is the proof of the probabilistic version of Proposition 3.

Proposition 3. *With probability* 1*, we project onto inactive constraints a finite number of times.*

8.3 Proof of part 2 of Theorem 3

The discussions in Iusem & De Pierro (1990); Iusem (1991) almost directly apply to that for our algorithm. For completeness, we present it along with the necessary modifications. As with the traditional Bregman algorithm, we first present the case when $f(x)$ is quadratic. That is,

$$f(x) = r + s^T \cdot x + \frac{1}{2} x^T H x$$

where H is a positive definite matrix. In this case, it is easy to see that

$$D_f(x, y) = \|x - y\|_H^2 := (x - y)^T H (x - y).$$

8.3.1 Proof of part 2 of Theorem 3—Quadratic Case

In this section, we will prove the following variation of Theorem 3.

Theorem 3. *If f is a strictly convex quadratic function, H_i are strongly zone consistent with respect to f, $x^0 = H^{-1} s \in S$, and the oracle $\mathcal{Q}$ satisfies either property 1 or 2, then there exists $\rho \in (0, 1)$ such that*

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^\nu\|_H} \leq \rho \tag{8.3}$$

where $\|y\|_H^2 = y^T H y$. In the case when we have an oracle that satisfies property 2, the limit in 8.3 holds with probability 1.

We establish some notation ahead of our lemmas. Let I be the set of all active constraints. That is, if x^* is the optimal solution then

$$I = \{i : \langle a_i, x^* \rangle = b_i\}.$$

Let S be the set of all x that satisfy these constraints (namely $S = \{x : \forall i \in I, \langle a_i, x \rangle = b_i\}$). Let H_x be the hyperplane, such that H_x represents the constraint in I that is furthest from x. Define

$$\mu = \inf_{x \notin S} \frac{d(x, H_x)}{d(x, S)}.$$

By Iusem & De Pierro (1990), we know that $\mu > 0$. Let U be the set of all optimal dual variables z; i.e., $U = \{z : \nabla f(x^*) = -A^T z\}$ and let $I_\nu = \{i : z_i^{\nu+1} \neq 0\}$.

Next, we present a few preliminary lemmas. These lemmas exist in some form or another in Iusem (1991); Iusem & De Pierro (1990) and we present them suitably modified for our purpose. These lemmas require the following set of assumptions about an iteration ν:

1. $\forall i \notin I$, $z_i^\nu = 0$;
2. for all $i \notin I$, we do not project onto this constraint in the νth iteration; and,
3. there exists $z \in U$, such that for all $i \notin I_\nu$, $z_i = 0$.

Lemma 1. *Let x^* be the optimal solution for an instance of problem 6.1. For any sequence $x^n \to x^*$ such that x^n, z^n maintain the KKT conditions, there exists an M, such that for all $\nu \geq M$, there exists a $z \in U$, such that for all $i \notin I_\nu$, we have that $z_i = 0$.*

Proof. Let $V_\nu = \{z : \forall i \notin I_\nu, z_i = 0\}$. Then assume, for the sake of contradiction, that the result is false. That is, there is a sequence ν_k such that $V_{\nu_k} \cap U = \emptyset$. Then since there finitely many different I_ν (hence finitely many V_ν), we have that one of these must occur infinitely often. Thus, by taking an appropriate subsequence, we assume, without loss of generality, that I_{ν_k} are all equal. Let $V = V_{\nu_k}$ and obtain $V \cap U = \emptyset$.

Since V is a closed subspace, U is a closed set, and $V \cap U = \emptyset$, we must have that $d(V, U) > 0$. But $z^{\nu_k+1} \in V$ and so

$$d(z^{\nu_k+C}, U) \geq d(V, U) > 0.$$

Since $x^\nu \to x^*$ and we maintain the KKT conditions, we have that for any $z \in U$,

$$A^T z^\nu = -\nabla f(x^\nu) \to -\nabla f(x^*) = A^T z.$$

Thus $d(z^\nu, U) \to 0$ which is a contradiction. □

Lemma 2. *For any sequence $x^n \to x^*$, if for a given ν, we have that the sequence satisfies assumptions (1) and (2), then*

$$\|x^{\nu+1} - x^*\|_Q^2 \leq \|x^\nu - x^*\|_Q^2 - \sum_{n=k}^{K} \|x^{n+1} - x^n\|_Q^2$$

where k and K are the indices of the first and last projection that take place in the νth iteration.

Proof. This Lemma is simply a statement about Bregman projections and so its proof requires no modification. □

Before we proceed, we introduce additional notation. Let A_{I_ν}, b_{i_ν} be the submatrix of A, b with rows from I_ν and

$$S_\nu = \{x : A_{I_{\nu_k}} x = b_{I_{\nu_k}}\}.$$

Lemma 3. *For any sequence such that $x^n \to x^*$, if for a given ν, we have that it satisfies assumptions (1), (2), and (3), then we have that $\|x^{\nu+1} - x^*\|_Q = d(x^{\nu+1}, S_\nu)$.*

Proof. Consider the constrained problem

$$\min_{x \in S_\nu} \|x^{\nu+1} - x\|_Q^2 \tag{8.4}$$

Then sufficient conditions for a pair (x, z_{I_ν}) to be optimal for this problem are

$$A_{I_\nu} x = b_{I_\nu} \text{ and } x = x^{\nu+1} - Q^{-1} A_{I_\nu}^T z_{I_\nu}$$

By Proposition 3, we see that since x^* is solution to problem 6.1, we have that $A_{I_\nu} x^* = b_{I_\nu}$. Then by assumptions and the manner in which we do projections, we have that there exists $z \in U$, such that for all $i \notin I_\nu$, $z_i = 0$ and

$$x^* = x^{\nu+1} - Q^{-1} A^T (z^{\nu+1} - z)$$

Then since $z_i^{\nu+1} = 0$ for all $i \notin I_\nu$, we have that

$$x^* = x^{\nu+1} - Q^{-1} A_{I_\nu}^T (z_{I_\nu}^{\nu+1} - z_{I_\nu})$$

Thus, x^* is the optimal solution to 8.4. □

Next for $x \notin S_\nu$, let H_x^ν be the hyperplane of that is furthest from x and define

$$\mu_\nu = \inf_{x \notin S_\nu} \frac{d(x, H_x^\nu)}{d(x, S_\nu)}$$

Now we are ready to prove the following theorem.

Theorem 5. *Let x^* is the optimal solution to problem 6.1. Then given ν that satisfies assumptions (1), (2), and (3), we have that*

$$\|x^{\nu+1} - x^*\|_Q^2 \leq \frac{L}{L+\mu^2}\|x^\nu - x^*\|_Q^2$$

where L is the number of projections that happened in νth iteration.

Proof. By Lemma 3, for any such ν we have that $x^{\nu+1} \notin S_\nu$ (or we have converged already). Suppose constraint $j \in I_\nu$ defines the hyperplane $H^\nu_{x^{\nu+1}}$. Then by Lemma 3 and definitions of μ_ν, μ we have the following inequality.

$$\begin{aligned}\|x^{\nu+1} - x^*\| &= d(x^{\nu+1}, S_\nu)\\ &\leq \frac{1}{\mu_\nu} d(x^{\nu+1}, H^\nu_{x^{\nu+1}})\\ &\leq \frac{1}{\mu} d(x^{\nu+1}, H^\nu_{x^{\nu+1}})\end{aligned}$$

Now since $I_\nu = \{i : z_i^{\nu+1} \neq 0\}$, we know that during the νth iteration we must have projected onto $H^\nu_{x^{\nu+1}}$. Note that this is the only place in the proof where we need the fact that we remember old constraints. Let us say that this happens during the rth projection of the νth iteration.

Note by assumption, we satisfy the assumptions of Lemma 2. Let $y^r, y^{\nu+1}$ be the projections of $x^r, x^{\nu+1}$ onto $H^\nu_{x^{\nu+1}}$. Then we see that

$$\begin{aligned}d(x^{\nu+1}, H^\nu_{x^{\nu+1}})^2 &= \|y^{\nu+1} - x^{\nu+1}\|_Q^2\\ &\leq \|y^r - x^{\nu+1}\|_Q^2 && [y^{\nu+1} \text{ by def is the closest point}]\\ &\leq \left(\|y^r - x^{r+1}\|_Q + \sum_{i=r+1}^{L}\|x^i - x^{i+1}\|_Q\right)^2 && [\text{Triangle inequality}]\\ &= \left(\sum_{i=r+1}^{L}\|x^i - x^{i+1}\|_Q\right)^2\\ &\leq \left(\sum_{i=0}^{L}\|x^i - x^{i+1}\|_Q\right)^2\\ &\leq L\sum_{i=0}^{L}\|x^i - x^{i+1}\|_Q^2 && [\text{ Cauchy Schwarz}]\\ &\leq L\left(\|x^\nu - x^*\|_Q^2 - \|x^{\nu+1} - x^*\|_Q^2\right) && [\text{Lemma 2}]\end{aligned}$$

Thus, we get that

$$\mu^2\|x^{\nu+1}x^*\|^2 \leq d(x^{\nu+1}, H^\nu_{x^{\nu+1}})^2 \leq L\left(\|x^\nu - x^*\|^2 - \|x^{\nu+1} - x^*\|_Q^2\right)$$

Rearranging, we get that

$$\|x^{\nu+1} - x^*\|_Q^2 \leq \frac{L}{L+\mu^2}\|x^\nu - x^*\|_Q^2.$$

□

As a corollary to the above theorem, we have that algorithm 1 converges linearly.

Theorem 3. *If f is a strictly convex quadratic function, H_i are strongly zone consistent with respect to f, $x^0 = H^{-1}s \in S$, and the oracle $\mathcal{Q}$ satisfies either property 1 or 2, then there exists $\rho \in (0, 1)$ such that*

$$\lim_{\nu\to\infty}\frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^\nu\|_H} \leq \rho \tag{8.3}$$

where $\|y\|_H^2 = y^T Hy$. In the case when we have an oracle that satisfies property 2, the limit in 8.3 holds with probability 1.

Proof. Using Proposition 3, Lemma 1, and that we have finitely many constraints, we see that if ν is large enough, the assumptions for Theorem 5 are satisfied. Taking the limit gives us the needed result.

In the case when we have an oracle that satisfies property 2, consider the product space of all possible sequences of hyperplanes returned by our oracle. In this product space, we see that with probability 1, we generate a sequence of hyperplanes, such that algorithm 1 converges. For any such sequence of hyperplanes, we have that 8.3 holds. Thus, the limit in 8.3 holds with probability 1 for random separation oracles. □

8.3.2 PROOF OF PART 2 OF THEOREM 3—GENERAL

The rate of convergence for the general Bregman method was established in Iusem (1991). To show this, let $\tilde{f}$ be the 2nd degree Taylor polynomial of f centered at the optimal solution x^*.

$$\tilde{f}(x) = f(x^*) + \nabla f(x^*)^T \cdot x + \frac{1}{2} x^T \cdot \nabla^2 f(x^*) \cdot x$$

For notational convenience, let H be the Hessian of f at x^*. Then we can see that if replace f with $\tilde{f}$ in 6.1 then the optimal solution does not change. Thus, if had access to $\tilde{f}$ and could use this function to do our projections, then from the quadratic case we have our result.

Thus, to get the general result, if x^ν is our standard iterate and $\tilde{x}^\nu$ is the iterate produced by using $\tilde{f}$ instead of f, then Iusem (1991) shows that $\|x^\nu - \tilde{x}^\nu\|$ is $o(\|x^\nu - x^*\|_H)$. Specifically, we can extract the following theorem from Iusem (1991).

Theorem 6. *Iusem (1991) Let x^* is the optimal solution for problem 6.1 and $\tilde{x}^n$ is the sequence produced by using the same sequence of hyperplanes but with $\tilde{f}$ instead of f. Given a sequence x^n produced by Bregman projections, such that $x^n \to x^*$, and for large enough ν we satisfy assumptions (1), (2), and (3), then $\|x^\nu - \tilde{x}^\nu\|$ is $o(\|x^\nu - x^*\|_H)$*

Using this we can get the general result as follows

$$\begin{aligned} \|x^{\nu+1} - x^*\|_H &\leq \|x^{\nu+1} - \tilde{x}^{\nu+1}\|_H + \|\tilde{x}^{\nu+1} - x^*\|_H \\ &\leq \|x^{\nu+1} - \tilde{x}^{\nu+1}\|_H + \rho\|\tilde{x}^\nu - x^*\|_H && \text{[Quadratic case convergence]} \\ &\leq \|x^{\nu+1} - \tilde{x}^{\nu+1}\|_H + \rho\|\tilde{x}^\nu - x^\nu\|_H + \rho\|x^\nu - x^*\|_H \end{aligned}$$

Then diving by $\|x^{\nu+1} - x^*\|_H$, and using Theorem 6 to take the limit, we get that there exists $\rho \in (0, 1)$ such that

$$\lim_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^\nu\|_H} \leq \rho \tag{8.3}$$

As with the quadratic case, we see that is an oracle satisfies property 2, then 8.3 holds with probability 1. Thus, we have proved Theorem 3 in its complete generality.

8.4 PROOF OF THEOREM 4

In this section we prove Theorem 4 in essentially the same manner as we did for Theorem 3 and so we outline only what changes are necessary.

Theorem 4. *If $f \in \mathcal{B}(S)$, the hyperplanes H_i are strongly zone consistent with respect to f, and $\exists\, x^0 \in S$ such that $\nabla f(x^0) = 0$, then with probability* 1 *any sequence x^n produced by the algorithm converges to the optimal solution of problem 6.1. Furthermore, if x^* is the optimal solution, f is twice differentiable at x^*, and the Hessian $H := Hf(x^*)$ is positive semidefinite, then there exists $\rho \in (0, 1)$ such that with probability* 1,

$$\liminf_{\nu \to \infty} \frac{\|x^* - x^{\nu+1}\|_H}{\|x^* - x^\nu\|_H} \leq \rho.$$

To prove Theorem 4, we need to analyze only what goes wrong if the algorithm “forgets” all of the old constraints. First, consider the proof in the case that we converge to the optimal solution. Then, steps 1,2, and 3 are completely unaffected by forgetting old constraints. The only step that is affected is step 4. In a previous proof, we argued that if for some inactive constraint a_i, z_i is non-zero infinitely often, then we projected onto this constraint infinitely often. In our present setting, we cannot conclude this directly as $z_i^\nu > 0$ does not imply that we remember a_i on the νth iteration. However, due to property 2, we know that $\mathcal{Q}$ returns a_i with probability at least τ. Thus, again using the Borel Cantelli Lemmas, we see that we have a_i infinitely often and this iteration converges to the optimal.

To prove the second part of the theorem, we recall from Theorem 6 that we only need to analyze the case when f is a quadratic function. Indeed, the only place where we used the fact that we remembered old constraints was in the proof of Theorem 5 in which we needed to remember old constraints to guarantee that during the νth iteration we project onto the constraint a_i that is furthest from x^ν among those constraints for which $z_i^{\nu+1} > 0$. We cannot guarantee that this happens always but we can guarantee that it happens infinitely often.

Therefore, the conclusion of Theorem 5 holds infinitely often instead of for the tail of the sequence and we replace the limit with a limit infimum to obtain the desired result.

9 Application Details

All code, data, and outputs from the experiments can be found at `https://www.dropbox.com/sh/lq5nnhi4je2lh89/AABUUW7k5z3lXTSm8x1hhN1Da?dl=0`. All code was written in Julia 1.1.0 and run on Google Cloud Compute instances.

9.1 Metric Nearness

Convergence Criterion.

One variant of the metric nearness problem is the decrease only variant, in which we are not allowed to increase the distances and must only decrease them. This problem can solved in $O(n^3)$ time by calculating the all pairs shortest path metric Gilbert & Jain (2017). Given x^n as input, let $\hat{x}^n$ be the optimal decrease only metric. We ran these experiments until $\|\hat{x}^n - x^n\|_2 \leq 1$.

Implementations.

We implemented the algorithm from Brickell et al. (2008). We made a small modification that improves the running time. In Brickell et al. (2008), it is recommended that we store the dual variable z as a sparse vector. However, as we do not want the overhead of handling sparse vectors, we store z as a dense vector.

Algorithm 1 was implemented with two modifications. As we can see from algorithm **??**, when the oracle finds violated constraints, it looks at each edge in G and then decides whether there is a violated inequality with that edge. It is cleaner in theory to find all such violated constraints at once and then do the project and forget steps. It is, however, much more efficient in practice to do the project and forget steps for a single constraint as we find it. This approach also helps cut down on memory usage.

The second modification is that once our oracle returns a list of constraints (note we have already projected onto these once), we project onto our whole list of constraints again. Thus, for the constraints returned by the oracle, we project onto these constraints twice per iteration. Note this does not affect any of the convergence results for the algorithm. The pseudocode for this modification can be seen in Algorithm 5.

Additional Test Case.

We also tested our algorithm on an additional type of random weighted complete graph. Let u_{ij} be sampled from the uniform distribution on $[0, 1]$ and v_{ij} from a standard normal, then the weight for an edge $e = ij$ is given by

$$w_{ij} = \lceil 1000 \cdot u_{ij} \cdot v_{ij}^2 \rceil$$

In this case, we got the following running times.

Algorithm 5 Pseudocode for the implementation for Metric Nearness.

```
$L^0 = \emptyset$, $z^0 = 0$. Initialize $x^0$ so that $\nabla f(x^0) = 0$.
while Not Converged do
    Let $d(i, j)$ be the weight of shortest path between nodes $i$ and $j$ or $\infty$ if none exists.
    $L = \emptyset$
    for Edge $e = i(, j) \in E$ do
        if $w(i, j) > d(i, j)$ then
            Let $P$ be the shortest path between $i$ and $j$.
            Let $C = P \cup \{(i, j)\}$.
            Project onto $C$ and update $x, z$.
            if $z_C! = 0$ then
                Add $C$ to $L$.
    $\tilde{L}^{\nu+1} = L^{\nu} \cup L$
    $x^{\nu+1}, z^{\nu+1} =$ Project($x^{\nu}, z^{\nu}, \tilde{L}^{\nu+1}$)
    $L^{\nu+1} =$ Forget($\tilde{L}^{\nu+1}$)
return $x$
```

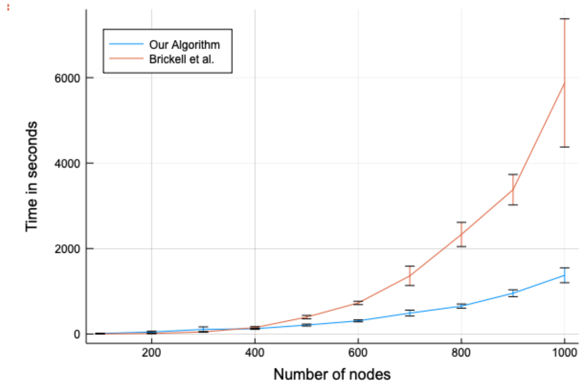

Figure 3: The red line is the mean running time for the algorithm from Brickell et al. (2008). The blue line is the running mean time for our algorithm. All computations were done on a machine with 4 physical cores, each with 13 GB of RAM.

9.2 Correlation Clustering

Transforming the LP. The formulation of the LP that we solve is as follows:

$$\begin{aligned} \text{minimize} \quad & \tilde{w}^T f + \frac{1}{\gamma} f^T \cdot W \cdot f \\ \text{subject to} \quad & x \in \text{MET}(G) \\ & f_{ij} = |x_{ij} - d_{ij}|, \quad (i, j) \in E. \end{aligned} \tag{9.1}$$

The transformation 2.2 into 9.1 has two parts. The first part is the transformation done in Veldt et al. (2019) to obtain the formulation presented in 4.1. To do this transformation we define $\tilde{w}(e) = |w^+(e) - w^-(e)|$. Then W is a diagonal matrix whose entries are given by $\tilde{w}$. Finally, we define d as follows

$$d_{ij} = \begin{cases} 1 & w^{-1}(e) > w^+(e) \\ 0 & \text{otherwise} \end{cases}$$

The second step of the transformation, is the relaxation from $x \in \text{MET}_n$ to $x \in \text{MET}(G)$. The proof for the second is now presented.

Proposition 2. *Let $f(x)$ be a function whose values only depends on the values x_{ij} for $e = (i, j) \in G$ and consider the following constrained optimization problem.*

$$\begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & x \in \textit{MET}(K_n) \end{array} \tag{9.2}$$

Let π be the projection from $\textit{MET}_n$ to $\textit{MET}(G)$ and let $\tilde{f}(\pi(x)) = f(x)$. Then for any optimal solution x^* to the following problem

$$\begin{array}{ll} \textit{minimize} & \tilde{f}(x) \\ \textit{subject to} & x \in \textit{MET}(G) \end{array} \tag{4.2}$$

we have that for all $\hat{x} \in \pi^{-1}(x^*)$, $\hat{x}$ is an optimal solution to 9.2.

Proof. Here, we see that if $\tilde{x}$ is the minimizer of 4.2 and x^* the minimizer of 9.2 then

$$f(x^*) = \tilde{f}(\pi(x^*)) \geq \tilde{f}(\tilde{x}) = f(\pi^{-1}(\tilde{x}))$$

□

Calculating the Approximation Ratio.

Let $\hat{x}$ be the optimal solution to 4.1, then if we let $R = \dfrac{\hat{x}^T W \hat{x}}{2\gamma \tilde{w}^T \hat{x}}$, by Veldt et al. (2019), we have that $\hat{x}$ is an $\dfrac{1+\gamma}{1+R}$ approximation to the optimal solution of 2.2. This is the formula we used to calculate the approximation ratios reported in Tables 1 and 2. For our experiments we used $\gamma = 1$.

Convergence Criterion.

We ran the experiment until the maximum violation of a metric inequality was at most 0.01. However, the two algorithms, Ruggles et al. (2019) and ours, have different metric constraints. Specifically Ruggles et al. (2019) only uses all constraints that come from 3 cycles, whereas we use all cycle constraints. Theoretically both sets of constraints define the same polytope, but practically there is a difference. Thus, in practice our algorithm was run to a slightly greater level of convergence than the one from Ruggles et al. (2019).

Implementations.

For the case when $G = K_n$, in addition to the modifications that were done for metric nearness experiment, we made two more modifications. First, we did the PROJECT and FORGET step one additional time per iteration. Second, we parallelized the oracle by running Dykstra's algorithm in parallel. The pseudocode for this version of algorithm 1 can be seen in Algorithm 6.

For the sparse version, we made only two modifications: we used the parallel version of the oracle, and during each iteration, we did the project and forget step 75 times.

Note that for both experiments, the additional constraints that were introduced due to the transformation were all projected onto once per iteration and never forgotten. The pseudocode for this version can be seen in Algorithm 7.

We used the implementation provided by the authors of Veldt et al. (2019) to run the experiments for their algorithm.

9.3 Information Theoretic Metric Learning

The hyper-parameters were set as follows: $\gamma = 1, u = 1, l = 10$. The pseudocode for our algorithm can be seen in Algorithm 8. The classification was done using the k nearest neighbor classifier.

Algorithm 6 Pseudocode for the implementation for CC for the dense case.

```
$L^0 = \emptyset$, $z^0 = 0$. Initialize $x^0$ so that $\nabla f(x^0) = 0$.
$L_a$ is the list of additional constraints. $z_a^0 = 0$ (dual for additional constraints)
**while** Not Converged **do**
    Let $d(i, j)$ be the weight of shortest path between nodes $i$ and $j$ or $\infty$ if none exists. This is found using a parallel algorithm.
    $L = \emptyset$
    **for** Edge $e = i(, j) \in E$ **do**
        **if** $w(i, j) > d(i, j)$ **then**
            Let $C = P \cup \{(i, j)\}$. Where $P$ be the shortest path between $i$ and $j$.
            Project onto $C$ and update $x, z$.
            **if** $z_C! = 0$ **then** Add $C$ to $L$.
    $L^\nu \leftarrow L^\nu \cup L$
    **for** $i = 1, 2$ **do**
        $x^\nu, z^\nu \leftarrow$ Project($x^\nu, z^\nu, L^\nu$)
        $L^\nu \leftarrow$ Forget($L^\nu$)
    $x^\nu, z_a^\nu \leftarrow$ Project($x^\nu, z_a^\nu, L_a$)
    $x^{\nu+1} = x^\nu, L^{\nu+1} = L^\nu$, $z^{\nu+1} = z^\nu$, $z_a^{\nu+1} = z_a^\nu$,
**return** $x$
```

Algorithm 7 Pseudocode for the implementation for CC for the sparse case.

```
$L^0 = \emptyset$, $z^0 = 0$. Initialize $x^0$ so that $\nabla f(x^0) = 0$.
$L_a$ is the list of additional constraints. $z_a^0 = 0$ (dual for additional constraints)
**while** Not Converged **do**
    Let $d(i, j)$ be the weight of shortest path between nodes $i$ and $j$ or $\infty$ if none exists. This is found using a parallel algorithm.
    $L = \emptyset$
    **for** Edge $e = i(, j) \in E$ **do**
        **if** $w(i, j) > d(i, j)$ **then**
            Let $C = P \cup \{(i, j)\}$. Where $P$ be the shortest path between $i$ and $j$.
            Add $C$ to $L$.
    $L^\nu \leftarrow L^\nu \cup L$
    **for** $i = 1, \dots, 75$ **do**
        $x^\nu, z^\nu \leftarrow$ Project($x^\nu, z^\nu, L^\nu$)
        $L^\nu \leftarrow$ Forget($L^\nu$)
    $x^\nu, z_a^\nu \leftarrow$ Project($x^\nu, z_a^\nu, L_a$)
    $x^{\nu+1} = x^\nu, L^{\nu+1} = L^\nu$, $z^{\nu+1} = z^\nu$, $z_a^{\nu+1} = z_a^\nu$,
**return** $x$
```

Algorithm 8 Pseudocode for the Project and Forget algorithm for ITML.

```
function PFITML($X, C, \gamma, u, l, S, D$)
    $\lambda^0 = 0$, $\Xi_{ij} = u$ for $(i, j) \in S$ and $\Xi_{ij} = l$ for $(i, j) \in D$. Initialize $C = I$.
    while Not Converged do
        Randomly sample $(i, j)$ from $S$
        Do projection for this constraint
        Randomly sample $(i, j)$ from $D$
        Do projection for this constraint
    return $C$
function PROJECTION($X, i, j, S, D, u, l, \Xi, \lambda, C$)
    $p = dist_C(X_i, X_j)$
    $\delta = 1$ if $(i, j) \in S$ and $\delta = -1$ if $(i, j) \in D$
    $\alpha = min\left(\lambda_{ij}, \frac{\delta}{2}\left(\frac{1}{p} - \frac{\gamma}{\Xi_{ij}}\right)\right)$
    $\beta = \frac{\delta\alpha}{1-\delta\alpha p}$
    $\Xi_{ij} = \gamma\frac{\Xi ij}{\gamma+\delta\alpha\Xi_{ij}}$
    $\lambda_{ij} = \lambda_{ij} - \alpha$
    $C = C + \beta C(x_i - x_j)(x_i - x_j j)^T C^T$
```