First-Order Algorithms for Optimization over Graph Laplacians

Tyler Maunu
Department of Mathematics
Brandeis University
Waltham, MA
maunu@brandeis.edu

***Abstract*—When solving an optimization problem over the set of graph Laplacian matrices, one must deal with a large number of constraints as well as the large objective variable size. In this paper we explore first-order methods for optimization over graph Laplacian matrices. These methods include two popular methods for constrained optimization: the mirror descent algorithm and the Frank-Wolfe (conditional gradient) algorithm. We derive efficiently implementable formulations of these algorithms over graph Laplacians, and use existing theory to show their iteration complexity in various regimes. Experiments demonstrate the efficiency of these methods over alternatives like interior point methods.**

I. Introduction

Working with graphs forms an increasingly important part of the toolkit for practitioners in data science and machine learning. The past few years have seen many new techniques for working with and processing graphs, from graph neural networks [Wu et al., 2020] to computational methods that explore the geometry and metric structure of the space of graphs [Gao et al., 2010, Chowdhury and Mémoli, 2019]. Study of graph properties is by no means new in data science, though, as it is fundamental in places like manifold learning and diffusion maps [Coifman et al., 2005, Coifman and Lafon, 2006] and Laplacian eigenmaps [Belkin and Niyogi, 2001].

The fundamental object of our study is the *graph Laplacian matrix*. We assume an undirected weighted graph $G(V, E, W)$, where $V = [n]$ is the set of nodes, $E \subset \{ij : i \in [n],\ j \in [n]\}$ is the set of edges, and $W = \{w_{ij} \in \mathbb{R}_{++} : ij \in E\}$ are the associated edge weights. The *weighted adjacency matrix* stores these weights in an $n \times n$ matrix

$$\boldsymbol{A}_{ij} = \begin{cases} w_{ij}, & ij \in E \\ 0, & ij \notin E. \end{cases} \tag{I.1}$$

The degree of a node is defined as the sum of the edge weights emanating from that node. Thus, for node i, the degree is given by

$$d_i = \sum_{j: ij \in E} w_{ij}. \tag{I.2}$$

One can compute the vector of degrees as $\boldsymbol{A}\boldsymbol{1} = \boldsymbol{d}$ and the *degree matrix* as $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{d})$. Putting these together, we finally arrive at the graph Laplacian, which is defined by the difference of these matrices,

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}. \tag{I.3}$$

The Laplacian is a fundamental object in graph theory. It's spectrum contains information about the connectivity of the graph [Belkin and Niyogi, 2001]. Furthermore, it can be used to define diffusion processes on graphs [Coifman and Lafon, 2006].

The goal of this paper is to explore optimization over the space of graphs from the perspective of convex optimization over graph Laplacians. While the idea of convex optimization over Laplacians is not new [Boyd, 2006, Dong et al., 2016, Mateos et al., 2019] and the algorithms we use are classical [Frank and Wolfe, 1956, Nemirovskij and Yudin, 1983], we offer a new perspective on the problem of optimization over graph Laplacians. In particular, we give the first principled study of first order methods for this problem. Our contributions are the following:

1) We derive Frank-Wolfe (FW) and Mirror Descent (MD) algorithms for optimization over fixed trace and fixed degree Laplacians. These algorithms also have simple updates.
2) We demonstrate how one can use results from the literature on convex optimization to give iteration complexity guarantees.

3) We run numerical experiments on a few examples to demonstrate the efficacy of these methods over existing methods.

We now outline the structure of our paper. In Section II we review related work. Then, in Section III, we derive our optimization algorithms and discuss their associated theoretical results. Finally, in Section IV, we run some experiments demonstrating the efficacy of our methods.

A. Notation

Let $\mathcal{H}_n$ denote the set of $n \times n$ real Hermetian matrices. We let $\mathbb{S}_+^d$ be the set of positive semidefinite (PSD) matrices. The Frobenius norm of a matrix is written as $\|\cdot\|_F$, and the matrix 1-norm is written as $\|\boldsymbol{A}\|_1 = \sum_{ij} |a_{ij}|$.

II. Review of Related Work

A series of recent works has focused on learning graph Laplacian matrices in a variety of practical settings [Xie et al., 2011, Dong et al., 2016, Kalofolias, 2016, Thanou et al., 2017, Egilmez et al., 2017, Pasdeloup et al., 2017, Segarra et al., 2017, Vlaski et al., 2018, Egilmez et al., 2018, Mateos et al., 2019, Kumar et al., 2019, Le Bars et al., 2019, Dong et al., 2019, Maretic and Frossard, 2020, Dong et al., 2020, Sahbi, 2021, Tugnait, 2021]. Most methods have focused on the development of energies that yield expressive Laplacian matrices from graph signals. For example, the Laplacian can be used to measure the smoothness of a graph signal. Dong et al. [2016] exploit this to recover network structures that are smooth in relation to observed signals. Other work has explored optimization of spectral quantities over Laplacian matrices [Boyd et al., 2004a, Boyd, 2006].

III. First-Order Optimization over Graph Laplacians

In this section we derive the main algorithms in this work. We begin in Section III-A with a discussion of the set of graph Laplacian matrices. We then discuss general convex programs in section III-B and derive the FW and MD algorithms for optimization over this set.

A. The Set of Graph Laplacian Matrices

As was mentioned, graphs can be encoded in matrix form by considering the weighted adjacency matrix or the graph Laplacian. Since we assume that the weights are symmetric and the graph is undirected, both $\boldsymbol{A}$ and $\boldsymbol{L}$ are symmetric. The graph Laplacian is so named because it acts as the analog of a second order differential operator on the underlying graph. For geometric graphs, where the weights are determined according to the Euclidean distance between points, the Laplacian converges to the Laplacian operator ∇^2 [Hein et al., 2007]. The Laplacian matrix also allows one to define a notion of smoothness on a graph. For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, which can be thought of as scalar measurements at each node, the product $\boldsymbol{L}\boldsymbol{x}$ takes the form $(\boldsymbol{L}\boldsymbol{x})_i = \sum_{j \in E_i} w_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)$, which is a local average difference. Furthermore, the quadratic form $\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}$ measures the Dirichlet energy, or smoothness of the signal $\boldsymbol{x}$. It is not hard to show from this that $\boldsymbol{L}$ must be positive semidefinite.

We can define the set of $n \times n$ weighted graph Laplacian matrices $\mathcal{L}_n$ by

$$\mathcal{L}_n = \{\boldsymbol{L} \in \mathcal{H}_n : \boldsymbol{L}_{ij} \geq 0 \text{ for } i = j, \quad \boldsymbol{L}_{ij} \leq 0 \text{ for } i \neq j, \ \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}\} \tag{III.1}$$

We can parametrize Laplacians in terms of their weighted adjacency matrix alone as $\boldsymbol{L} = \operatorname{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}$, a fact which we will use later in our optimization algorithms. Finally, we define the set of fixed-trace graph Laplacians $\mathcal{L}_n(T) = \{L \in \mathcal{L}_n : \operatorname{Tr}(L) = T\}$ and the set of Laplacians with fixed degree as $\mathcal{L}_n(\boldsymbol{d}) = \{L \in \mathcal{L}_n : \operatorname{diag}(\boldsymbol{L}) = \boldsymbol{d}\}$.

B. Convex Optimization over Laplacians

We consider first-order methods to optimize over Laplacians. We consider general functions $F : \mathcal{L}_n \to \mathbb{R}$. Suppose that we have a Euclidean convex function $F : \mathbb{R}^{n \times n} \to \mathbb{R}$ (which is convex over $\mathcal{L}_n$ since $\mathcal{L}_n$ is a Euclidean convex set). Denote the map π by

$$\pi(\boldsymbol{A}) = \operatorname{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}. \tag{III.2}$$

We have the following lemma, which states that this function must also be convex over the weighted adjacency matrix.

Lemma 1. *If F is convex over $\mathcal{L}_n$, then $F \circ \pi$ is convex over the set of weighted adjacency matrices.*

We now outline our two methods to solve convex optimization problems over $\mathcal{L}_n$: the FW and MD algorithms.

1) Frank-Wolfe: The FW method maintains feasibility of the iterates by using the iteration

$$\boldsymbol{L}_{k+1} = (1 - \eta_k)\boldsymbol{L}_k + \eta_k \boldsymbol{G}_k, \tag{III.3}$$

$$\boldsymbol{G}_k = \arg\min_{\boldsymbol{S} \in \mathcal{L}_n(\cdot)} \operatorname{Tr}(\nabla F(\boldsymbol{L}_k)\boldsymbol{S}). \tag{III.4}$$

Here $\boldsymbol{G}_k$ is called a *conditional gradient*, which is given by the solution to a certain linear program, and $\eta_k \in (0,1)$ is the step size. Due to the issues discussed in the appendix, the FW linear program is well-posed when the optimization considered is over fixed trace or fixed degree Laplacian matrices, which is why we write the optimization problem in (III.4) over $\mathcal{L}_n(\cdot)$.

Theorem 2. *Let $\boldsymbol{L}_*$ be a minimizer of a convex F with β-Lipschitz gradient. If we use the FW iteration to optimize it over $\mathcal{L}_n(\cdot)$, then*

$$F(\boldsymbol{L}_k) - F(\boldsymbol{L}_*) \leq \frac{4\beta D^2}{k+\xi}, \tag{III.5}$$

Furthermore, if F is strongly convex and $\boldsymbol{L}_*$ does not lie on the boundary of $\mathcal{L}_n(\cdot)$ and line search is used to select the step size η_k, then

$$F(\boldsymbol{L}_k) - F(\boldsymbol{L}_*) \leq \exp\left(-k\frac{\alpha\epsilon}{\beta D}\right)(F(\boldsymbol{L}_0) - F(\boldsymbol{L}_*)), \tag{III.6}$$

here 2ϵ is the distance from $\boldsymbol{L}_*$ to the boundary of $\mathcal{L}_n(\cdot)$ and D is the diameter of $\mathcal{L}_n(\cdot)$.

More general linear convergence results can be proved when $\boldsymbol{L}_*$ lies on the boundary using other variants of FW – see the variants discussed in Lacoste-Julien and Jaggi [2015], which can all be adapted to the Laplacian case as well. We illustrate this in the following.

At each iteration, one also computes an away step using the solution to the linear program

$$\max_{\boldsymbol{S}\in\mathcal{S}} \operatorname{Tr}(\nabla F(\boldsymbol{L}_k)\boldsymbol{S}), \tag{III.7}$$

where $\mathcal{S} \subset V(\mathcal{L}_n(\cdot))$ is a subset of the vertices of $\mathcal{L}_n$.

2) Mirror Descent: In the previous section, due to the ill-posedness of the linear program for general Laplacians, we had to restrict the FW method to operate on fixed-trace or fixed-degree problems. However, it may be the case that one does not know the degree or trace of the desired Laplacian. To offer an alternative that can handle variable trace and also achieves different theoretical guarantees, we consider MD.

We consider mirror descent on the adjacency matrix through the composition $F(\boldsymbol{L}) = F(\pi(\boldsymbol{A}))$. The mirror map we use is then the off diagonal entropy $\Phi(\boldsymbol{A}) = H_{od}(\boldsymbol{A})$. With this choice of mirror map, the MD update is then defined by

$$\begin{aligned}\boldsymbol{A}_{k+1} &= \exp\left(\log(\boldsymbol{A}_k) - \eta\nabla F(\boldsymbol{A}_k)\right) \\ &= \boldsymbol{A}_k \exp\left(-\eta\nabla F(\pi(\boldsymbol{A}_k))\right).\end{aligned} \tag{III.8}$$

where it is assumed that the exp and log are applied elementwise (i.e., they are *not* the matrix exponential and logarithm). If MD is applied in the fixed trace case, the update is given by

$$\boldsymbol{A}_{k+1} = T\frac{\boldsymbol{A}_k \odot \exp\left(-\eta\nabla F(\boldsymbol{A}_k)\right)}{\|\boldsymbol{A}_k \odot \exp\left(-\eta\nabla F(\boldsymbol{A}_k)\right)\|_1}, \tag{III.9}$$

where it is assumed that the diagonal remains fixed as 0 throughout the iteration. Similarly, the fixed degree case can be solved via a Sinkhorn style projection after each iteration. We note that these iterations are essentially the extension of exponential weights to the case of weighted graph adjacency matrices.

We have the following convergence theorem for mirror descent from Bubeck et al. [2015], Radhakrishnan et al. [2020].

Theorem 3. *Let L_* be the minimizer of a function F that is convex and has β-Lipschitz gradient with respect to $\|\cdot\|$, and let Φ be a mirror map ρ-strongly convex with respect to $\|\cdot\|$, for some norm $\|\cdot\|$. Then,*

$$F(\bar{\boldsymbol{L}}) - F(\boldsymbol{L}_*) \leq \frac{R^2\beta}{\rho t}. \tag{III.10}$$

Finally, if F is α-strongly convex and ∇F is β-Lipschitz with respect to $\|\cdot\|$, then mirror descent achieves a linear rate

$$F(\boldsymbol{L}_k) - F(\boldsymbol{L}_*) \leq \exp\left(-k\frac{\alpha\rho}{\beta L}\right)(F(\boldsymbol{L}_0) - F(\boldsymbol{L}_*)). \tag{III.11}$$

We remark that this also has an extension to convex Lipschitz without the assumption of Lipschitz gradient.

We give a summary the proposed methods below. In the implementation of the Frank-Wolfe algorithm for the fixed degree case, we note that one must solve an optimal-transport like linear program at each iteration. This can either be solved with a standard LP solver, or approximately solved by Sinkhorn's algorithm Cuturi [2013]. On the other hand, for mirror descent, the fixed degree case can be explicitly written as a certain Sinkhorn method applied to the updated adjacency matrix to reweight the rows and columns to have the correct degree. This is stated formally in the following lemma.

3) A Comment on Projected Gradient Descent: Projected gradient descent is a potential method to optimize over graph Laplacians as well, and this can be seen in past works such as Boyd et al. [2004b], Dong and Sawin [2020]. However, projection onto the set of

Set	Update Direction
$\mathcal{L}_n$	Undefined
$\mathcal{L}_n(T)$	$\mathcal{I} = \text{argmax}_{ij}\, \boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - \boldsymbol{C}_{ij} - \boldsymbol{C}_{ji}$ $\boldsymbol{A}_{ij} = a_{ij},\ ij \in \mathcal{I},\ a_{ij} \geq 0$ $\sum_{ij} \boldsymbol{A}_{ij} = T$
$\mathcal{L}_n(\boldsymbol{d})$	$\underset{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \text{diag}(\boldsymbol{A})=\boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{1}=\boldsymbol{d}}}{\text{argmin}} \langle \boldsymbol{A}, -\boldsymbol{C}\rangle$

TABLE I
FRANK-WOLFE ALGORITHMS.

Set	Update
$\mathcal{L}_n$	$\boldsymbol{A}_{k+1} = \boldsymbol{A}_k \odot \exp\left(-\eta \nabla F(\boldsymbol{A}_k)\right)$
$\mathcal{L}_n(T)$	$\boldsymbol{A}_{k+1} = T \frac{\boldsymbol{A}_k \odot \exp(-\eta\nabla F(\boldsymbol{A}_k))}{\|\boldsymbol{A}_k \odot \exp(-\eta \nabla F(\boldsymbol{A}_k))\|_1}$
$\mathcal{L}_n(\boldsymbol{d})$	$\mathsf{Sinkhorn}((\boldsymbol{A}_k \odot \exp\left(-\eta\nabla F(\boldsymbol{A}_k)\right)), \boldsymbol{d})$

TABLE II
MIRROR DESCENT ALGORITHMS.

Laplacians for general norms does not have a closed form solution. In fact, only projection with respect to the 1-norm has a closed form [Sato, 2019]. To project with respect to the Frobenius norm, one must either result to solving a complicated quadratic program or use other approximations.

IV. APPLICATIONS AND EXPERIMENTS

We now consider applications of our developed algorithms. In Section IV-A, we consider solving the Euclidean projection onto the set of Laplacian matrices using FW and MD. Then, in Section IV-B, we consider recovery of a Laplacian matrix from reduced measurements, which arises in the context of compressed sensing. All experiments are run on a 2020 Macbook Pro with a 2 GHz Quad-Core Intel Core i5 CPU and 16 GB RAM.

A. Laplacian Projection

Suppose that we have a matrix $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ that we wish to project to the space of graph Laplacians. The following convex program can be solved for this task:

$$\min_{\boldsymbol{L}\in\mathcal{L}_n} \|\boldsymbol{S} - \boldsymbol{L}\|, \tag{IV.1}$$

where $\|\cdot\|$ is some choice of norm. Having such a projection is essential for the implementation of projected gradient methods such as those in Boyd et al. [2004b], Dong and Sawin [2020]. We could in theory use out of the box convex solvers for this problem. For example, if the norm chosen is the Frobenius norm, then (IV.1) a quadratic program, but these are expensive to solve in general. On the other hand, first order methods like FW take $O(n^2)$ per iteration, and they converge linearly in certain cases, and so to achieve error ϵ they take $O(n^2 \log(1/\epsilon)$ complexity in this best case.

We show an example in Figure 6 where we use the MD and FW algorithms to solve a Laplacian projection problem with respect to the Frobenius norm. A Laplacian drawn from the stocastic block model is perturbed by small Gaussian noise, and we then try to project it back to the set of Laplacians. We see that MD converges extremely quickly to the projected matrix.

B. Synthetic Experiments: Laplacian Recovery

We now discuss the recovery of an underlying graph Laplacian matrix $\boldsymbol{L}$ from quadratic observations. Solving the structured matrix recovery problem has been present in many past works. We consider observations $(\boldsymbol{a}_i, y_i)$, where

$$\mathcal{A}(\boldsymbol{L})_i = y_i = \boldsymbol{a}_i^T \boldsymbol{L} \boldsymbol{a}_i,\ i = 1, \ldots, n. \tag{IV.2}$$

Two natural questions arise based on this sensing model: how many measurements are needed to specify $\boldsymbol{L}$ with high probability? What efficient optimization methods exist to recover $\boldsymbol{L}$? In this paper we focus on the latter question.

We consider two convex optimization programs for recovering $\boldsymbol{L}$. The first functional is standard least squares following Chen et al. [2015], Cai and Zhang [2015]

$$\min_{\boldsymbol{M}\in\mathcal{L}_n} \frac{1}{m}\sum_{i=1}^{n} (y_i - \boldsymbol{a}_i^T \boldsymbol{M} \boldsymbol{a}_i)^2 =: F_{LS}(\boldsymbol{M}). \tag{IV.3}$$

The second is the functional inspired by the Bures-Wasserstein metric [Maunu et al., 2023],

$$\min_{\boldsymbol{M}\in\mathcal{L}_n} \sum_{i=1}^{n} (\sqrt{y_i} - \sqrt{\boldsymbol{a}_i^T \boldsymbol{M} \boldsymbol{a}_i})^2 =: F_{BW}(\boldsymbol{M}). \tag{IV.4}$$

Under certain conditions, we expect these functions to be strongly convex [Chen et al., 2015, Maunu et al., 2023]. Furthermore, if the underlying matrix does not lie on the boundary of $\mathcal{L}_n(\cdot)$, we expect linear convergence. We observe this to be the case in Figures 1 and 2.

C. Real Data Experiments

Here we test out our algorithm on real data. We consider a similar experiment to that in Dong et al. [2016], where we take daily temperature at various locations on the east coast for the years of 2006-2020 [NCEI, 2020]. At the twenty cities, we observe an average temperature

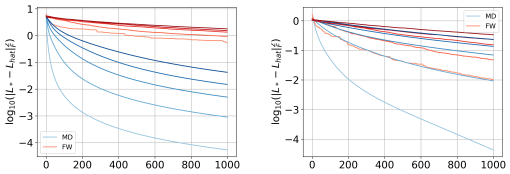

Fig. 1. Convergence on recovery problem in sublinear and linear convergence settings with least squares cost (IV.3) (depending on if the optimal point lies on the boundary). Here, the graph size 10, 20, 30, 40, 50. The darker shade corresponds to a larger graph.

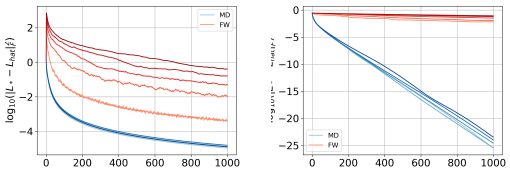

Fig. 2. Convergence on a recovery problem in sublinear and linear convergence setting (depending on it the optimal point lies on the boundary) with the Bures-Wasserstein cost (IV.4). Here, the graph size 10, 20, 30, 40, 50. The darker shade corresponds to a larger graph.

each day. We denote the vector of temperatures on the tth day as $\boldsymbol{x}_t$, for $t = 1, \ldots, 365$. In Figure 3 we display the temperatures by day.

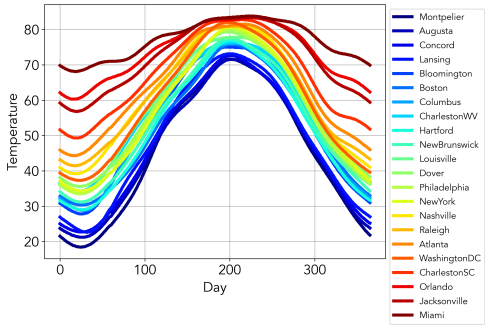

Fig. 3. Caption

In the first experiment, we seek to learn a graph such that the signals $\boldsymbol{x}_t$ are smooth. This is accomplished with an energy

$$\min_{\boldsymbol{M} \in \mathcal{L}_n(20)} \sum_{t=1}^{T} (\boldsymbol{x}_t^\top \boldsymbol{M} \boldsymbol{x}_t - 0.05)^2 \tag{IV.5}$$

In the second experiment, we incorporate a temporal aspect. One can write a graph notion of diffusion with

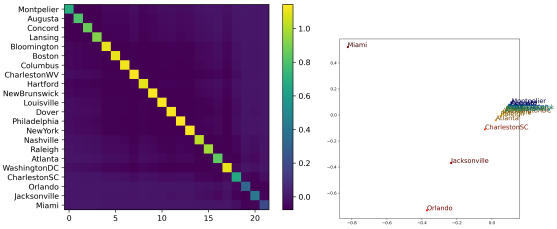

Fig. 4. Recovery with the smoothness energy (IV.5).

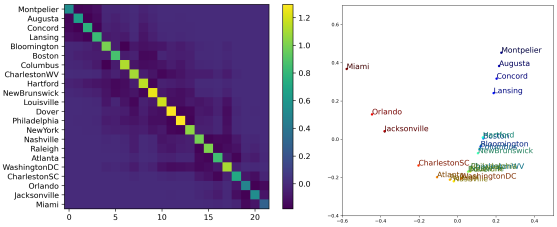

Fig. 5. Recovery with the diffusion energy (IV.8). Points are embedded with a two-dimensional Laplacian eigenmap.

the differential equation [Thanou et al., 2017]

$$\frac{d\boldsymbol{x}}{dt} = -c\boldsymbol{L}\boldsymbol{x}, \tag{IV.6}$$

where $\boldsymbol{L}$ is a graph Laplacian and $\boldsymbol{x}$ is the time dependent graph signal. In our dataset, we observe discrete derivatives

$$\left.\frac{d\boldsymbol{x}}{dt}\right|_{t=s} \approx \boldsymbol{x}_{s+1} - \boldsymbol{x}_s, \tag{IV.7}$$

and we try to solve the optimization problem

$$\min_{\boldsymbol{A} \in \mathcal{L}_n(20)} \sum_{t=1}^{T-1} \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t + \boldsymbol{A}\boldsymbol{x}_t\|^2 \tag{IV.8}$$

The results of these two energies are displayed in Figures 4 and 5.

V. Conclusions

In this paper, we have presented some new algorithms for solving optimization problems over the set of graph Laplacian matrices. In particular, we have derived Frank-Wolfe and mirror descent algorithms for general costs. These algorithms are easily implementable, and we note that to our knowledge these are the first implementations of them on graph Laplacians. Our simulations verify our theory and point to settings where the methods converge both linearly and sublinearly.

REFERENCES

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

Stephen Boyd. Convex optimization of graph laplacian eigenvalues. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 1311–1319. Citeseer, 2006.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004a.

Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 46(4): 667–689, 2004b.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

T Tony Cai and Anru Zhang. ROP: matrix recovery from rank one projections. *The Annals of Statistics*, 43(1): 102–138, 2015.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Yuxin Chen, Yuejie Chi, and Andrea J Goldsmith. Exact and stable covariance estimation from quadratic sampling via convex programming. *IEEE Transactions on Information Theory*, 61(7):4034–4059, 2015.

Samir Chowdhury and Facundo Mémoli. The gromov–wasserstein distance between networks and stable network invariants. *Information and Inference: A Journal of the IMA*, 8(4):757–787, 2019.

Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1): 5–30, 2006.

Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431, 2005.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal processing magazine*, 37 (6):117–127, 2020.

Yihe Dong and Will Sawin. Copt: Coordinated optimal transport on graphs. *Advances in Neural Information Processing Systems*, 33:19327–19338, 2020.

Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from filtered signals: Graph system and diffusion kernel identification. *IEEE Transactions on Signal and Information Processing over Networks*, 5 (2):360–374, 2018.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010.

Jacques Guélat and Patrice Marcotte. Some comments on wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986.

Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8(6), 2007.

Vassilis Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929. PMLR, 2016.

Sandeep Kumar, Jiaxi Ying, José Vinícius de Miranda Cardoso, and Daniel Palomar. Structured graph learning via laplacian spectral constraints. *Advances in neural information processing systems*, 32, 2019.

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. *Advances in neural information processing*

systems, 28, 2015.

Batiste Le Bars, Pierre Humbert, Laurent Oudre, and Argyris Kalogeratos. Learning laplacian matrix from bandlimited graph signals. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2937–2941. IEEE, 2019.

Hermina Petric Maretic and Pascal Frossard. Graph laplacian mixture model. *IEEE Transactions on Signal and Information Processing over Networks*, 6:261–270, 2020.

Gonzalo Mateos, Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro. Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine*, 36(3):16–43, 2019.

Tyler Maunu, Thibaut Le Gouic, and Philippe Rigollet. Bures-wasserstein barycenters and low-rank matrix recovery. *(to appear) AISTATS*, 2023.

NCEI. U.s. climate normals 2020: U.s. daily climate normals (2006-2020, 2020. URL https://www.ncei.noaa.gov/access/search/dataset-search.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Bastien Pasdeloup, Vincent Gripon, Grégoire Mercier, Dominique Pastor, and Michael G Rabbat. Characterization and inference of graph diffusion processes from observations of stationary signals. *IEEE transactions on Signal and Information Processing over Networks*, 4(3):481–496, 2017.

Adityanarayanan Radhakrishnan, Mikhail Belkin, and Caroline Uhler. Linear convergence of generalized mirror descent with time-dependent mirrors. *arXiv preprint arXiv:2009.08574*, 2020.

Hichem Sahbi. Learning laplacians in chebyshev graph convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2064–2075, 2021.

Kazuhiro Sato. Optimal graph laplacian. *Automatica*, 103:374–378, 2019.

Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. Network topology inference from spectral templates. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):467–483, 2017.

Dorina Thanou, Xiaowen Dong, Daniel Kressner, and Pascal Frossard. Learning heat diffusion graphs. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):484–499, 2017.

Jitendra K Tugnait. Sparse graph learning under laplacian-related constraints. *IEEE Access*, 9:151067–151079, 2021.

Stefan Vlaski, Hermina P Maretić, Roula Nassif, Pascal Frossard, and Ali H Sayed. Online graph learning from sequential data. In *2018 IEEE Data Science Workshop (DSW)*, pages 190–194. IEEE, 2018.

Philip Wolfe. Convergence theory in nonlinear programming. *Integer and nonlinear programming*, pages 1–36, 1970.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Bo Xie, Meng Wang, and Dacheng Tao. Toward the optimization of normalized graph laplacian. *IEEE transactions on neural networks*, 22(4):660–666, 2011.

Mingrui Zhang, Zebang Shen, Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. One sample stochastic frank-wolfe. In *International Conference on Artificial Intelligence and Statistics*, pages 4012–4023. PMLR, 2020.

APPENDIX

A. Supplementary Proofs

1) Refresher on Euclidean Optimization: Suppose that we wish to solve

$$\min_{x \in \mathcal{C}} f(x), \tag{A.1}$$

where $\mathcal{C}$ is a subset of $\mathbb{R}^d$. Two popular methods to solve this problem in practice are the Frank-Wolfe method and the mirror descent method. In both cases, one relies on an efficiently implementable update.

2) Frank-Wolfe Method: The Frank-Wolfe, or conditional gradient descent method, over a Euclidean space, maintains feasibility by solving a linear program at each iteration. In particular, Frank-Wolfe iterates by solving

$$x_{k+1} = (1 - \eta_k)x_k + \eta_k g_k, \tag{A.2}$$

$$g_k = \operatorname*{argmin}_{g \in \mathcal{C}} \langle g, \nabla f(x_k) \rangle. \tag{A.3}$$

Here, if $g_k = \nabla f(x_k)$. then we refer to the method as the Frank-Wolfe (FW) algorithm. On the other hand, if one uses a stochastic approximation to the gradient for g_k, as is common in modern optimization, we refer to the method as the Stochastic Frank-Wolfe (SFW) algorithm.

We have the following standard rate of convergence. For a proof, see Guélat and Marcotte [1986].

Theorem 4. *Suppose that f is convex and has β-Lipschitz gradient and $\eta_k = \frac{2}{2+k}$. Then,*

$$f(x_k) - f(x_*) \leq \frac{L}{k + \xi}, \tag{A.4}$$

for some constant ξ. If it is further assumed that f is α-strongly convex and x_* is the in the relative interior of $\mathcal{C}$, then x_k converges linearly to x_*.

Notice that even in the strongly convex case, if the optimal point lies on the boundary, then convergence is slow. To address this, Wolfe [1970] introduced the away step variant of Frank-Wolfe, which uses an active set of vertices to help the method move in directions that are less parallel to the boundary (i.e., it deals with the zig-zagging phenomenon). In Lacoste-Julien and Jaggi [2015], the authors study this and some other variants of Franke-Wolfe and prove linear convergence in the strongly convex setting.

In the stochastic setting, one must use other strategies to ensure convergence. Another popular idea involves average the gradient at each iteration before solving the linear program Zhang et al. [2020]:

$$g_t = (1 - \eta_t)(g_{t-1} + \tilde{\Delta}_t) + \eta_t \nabla \tilde{f}(x_t), \tag{A.5}$$

where $\nabla \tilde{f}(x_t)$ is the stochastic approximation to the gradient of f and $\tilde{\Delta}_t$ is an unbiased estimator of $\nabla f(x_t) - \nabla f(x_{t-1})$. These methods typically converge at a $1/\sqrt{t}$ sublinear rate in the convex setting and a $1/t$ sublinear rate in the strongly convex setting.

3) Mirror Descent: Over Euclidean space, given a strictly convex function Φ, the Bregman divergence is given by

$$D_\Phi(x, y) = \Phi(x) - \Phi(y) - \langle \nabla \Phi(y), x - y \rangle. \tag{A.6}$$

By convexity, $D_\Phi \geq 0$.

The mirror descent (MD) iteration [Beck and Teboulle, 2003] for a function f is

$$x_{k+1} = \operatorname*{argmin}_{x \in \mathcal{C}} \langle \eta \nabla f(x_k), x - x_k \rangle + D_\Phi(x, x_k). \tag{A.7}$$

Solving the minimization yields the other familiar MD iteration

$$\nabla \Phi(x_{k+1}) = \nabla \Phi(x_k) - \eta \nabla f(x_k). \tag{A.8}$$

One popular choice of mirror map is the entropy $\Phi(x) = \sum_i x_i \log x_i$. In this case, the optimization algorithm over the simplex is the exponential weights algorithm [Cesa-Bianchi and Lugosi, 2006]. Indeed, it is not hard to see that

$$\nabla \Phi(x) = (\log x_i + 1)_{i=1}^n, \ \nabla \Phi^*(y) = (e^{y_i - 1})_{i=1}^n \tag{A.9}$$

yields the update

$$x_{k+1} = \exp(\log x_k - \eta \nabla f(x_k)) = x_k \exp(-\eta \nabla f(x_k)) \tag{A.10}$$

If the optimization is constrained over the simplex, where the elements of x_k must sum to one, an additional normalization is considered as $x_{k+1} = x_{k+1}/\|x_{k+1}\|_1$.

B. Laplacian Linear Programs

We begin by consider linear programs over graph Laplacians. This is used as a subroutine in the Frank-Wolfe algorithm. In its most general form, a linear program over graph Laplacians would take the form

$$\min_{\boldsymbol{L} \in \mathcal{L}_n} \langle \boldsymbol{L}, \boldsymbol{C} \rangle. \tag{A.11}$$

Notice that this can be written as a linear program in the variable $\boldsymbol{A}$ by

$$\min_{\operatorname{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A} \in \mathcal{L}_n} \langle \operatorname{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A}, \boldsymbol{C} \rangle \tag{A.12}$$

$$\equiv \min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\boldsymbol{0}}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}, \boldsymbol{C}\rangle$$

However, this problem may be ill-posed in general without additional constraints. Therefore, we consider optimization over fixed trace and fixed degree Laplacian matrices

$$\min_{\boldsymbol{L}\in\mathcal{L}_n(T)} \langle \boldsymbol{L}, \boldsymbol{C}\rangle \equiv \min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\boldsymbol{0} \\ \sum_{ij}\boldsymbol{A}_{ij}=T}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}, \boldsymbol{C}\rangle. \tag{A.13}$$

$$\min_{\boldsymbol{L}\in\mathcal{L}_n(\boldsymbol{d})} \langle \boldsymbol{L}, \boldsymbol{C}\rangle \equiv \min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{1}=\boldsymbol{d}}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}, \boldsymbol{C}\rangle. \tag{A.14}$$

We also define some regularized surrogate convex programs using an entropic regularizer. The use of an entropic regularizer has become popular in linear programming to find solutions to problems such as optimal transport Cuturi [2013]. We define the off-diagonal entropy as

$$H_{od}(\boldsymbol{P}) = \sum_{i\neq j} \boldsymbol{P}_{ij}(\log \boldsymbol{P}_{ij} - 1). \tag{A.15}$$

and consider the surrogate convex programs which add $-\lambda H$ to the objectives (A.13) and (A.14).

$$\min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\boldsymbol{0} \\ \sum_{ij}\boldsymbol{A}_{ij}=T}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}, \boldsymbol{C}\rangle - \lambda H_{od}(\boldsymbol{A}), \tag{A.16}$$

$$\min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{1}=\boldsymbol{d}}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}, \boldsymbol{C}\rangle - \lambda H_{od}(\boldsymbol{A}). \tag{A.17}$$

We collect all of the solutions for the Laplacian linear programs in Table III. We see that the fixed trace linear programs have closed form solutions while the fixed degree linear programs must be solved using other algorithms. These solutions are derived in the appendix.

1) Results in Table III: We first discuss solutions to (A.13) and its regularized variant.

Lemma 5. *The entropically regularized variant of* (A.13) *has solution* $T\boldsymbol{A}/\|\boldsymbol{A}\|_1$, *where*

$$\boldsymbol{A}_{ij} = \sum_{ij} \exp\left(\frac{1}{2\epsilon}(\boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - \boldsymbol{C}_{ij})\right) \tag{A.18}$$

Furthermore, define the set

$$\mathcal{J} = \operatorname*{argmax}_{ij} \boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - \boldsymbol{C}_{ij} - \boldsymbol{C}_{ji}. \tag{A.19}$$

Formulation	Solution
Fixed Trace Ent. Reg.	$\boldsymbol{A}_{ij} = \sum_{ij} \exp\left(\frac{1}{2\epsilon}(\boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - \boldsymbol{C}_{ij})\right)$ $\sum_{ij}\boldsymbol{A}_{ij} = T$
Fixed Trace	$\mathcal{J} = \mathrm{argmax}_{ij}\, \boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - \boldsymbol{C}_{ij} - \boldsymbol{C}_{ji}$ $\boldsymbol{A}_{ij} = a_{ij},\ ij \in \mathcal{J},\ a_{ij} \geq 0$ $\sum_{ij}\boldsymbol{A}_{ij} = T$
Fixed Degree Ent. Reg	Sinkhorn [Cuturi, 2013]
Fixed Degree	Interior Point Method
No Constraints	Ill-posed

TABLE III
SOLUTIONS TO REGULARIZED AND UNREGULARIZED LAPLACIAN LINEAR PROGRAMS.

The solution to the linear program (A.11) *is any matrix* $\boldsymbol{B} = T\boldsymbol{A}/\|\boldsymbol{A}\|_1$, *where*

$$\boldsymbol{A}_{ij} = \begin{cases} a_{ij}, & ij \in \mathcal{J} \\ 0, & else. \end{cases} \tag{A.20}$$

where $a_{ij} \geq 0$, *with at least one* $ij \in \mathcal{J}$ *such that* $a_{ij} > 0$.

For the degree constrained case as well as its entropically regularized variants, we can follow the literature on optimal transport and derive a Sinkhorn style algorithm to solve an entropically regularized version of this problem.

In other words, we instead propose to solve the surrogate problem

$$\min_{\substack{\boldsymbol{A}\boldsymbol{1}=\boldsymbol{d} \\ \boldsymbol{A}=\boldsymbol{A}^T}} \langle \boldsymbol{A}, -\boldsymbol{C}\rangle - \epsilon H_{od}(\boldsymbol{A}) + \mathbb{1}\left(\mathrm{diag}(\boldsymbol{A}) = 0\right). \tag{A.21}$$

Examining the KKT conditions are

$$\boldsymbol{A}_{ij} = \exp\left(\frac{1}{\epsilon}(\boldsymbol{C}_{ij}) + \boldsymbol{f}_i + \boldsymbol{g}_j\right),\ i \neq j, \tag{A.22}$$
$$\boldsymbol{A}\boldsymbol{1} = \boldsymbol{d}.$$

which is what one would get if one considered an entropic regularization of the second version of the linear program. In any case, we can solve the entropically regularized Laplacian linear program with fixed degree by appealing to the Sinkhorn algorithm [Cuturi, 2013]. We refer to the resulting method as Fixed Degree Laplacian Sinkhorn.

2) Connection Between Laplacian Linear Programs and Optimal Transport: In this case, the linear program is equivalent to

$$\min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{1}=\boldsymbol{d}}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}, \boldsymbol{C}\rangle$$

$$= \langle \mathrm{diag}(\boldsymbol{d}), \boldsymbol{C} \rangle + \min_{\substack{\boldsymbol{A} \in \mathcal{H}_n \\ \boldsymbol{A} \geq 0,\ \mathrm{diag}(\boldsymbol{A}) = \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{1} = \boldsymbol{d}}} \langle \boldsymbol{A}, -\boldsymbol{C} \rangle. \quad \text{(A.23)}$$

While we could employ various linear programming techniques to solve this problem.

$$\min_{\substack{\boldsymbol{A} \in \mathcal{H}_n \\ \boldsymbol{A} \geq 0,\ \mathrm{diag}(\boldsymbol{A}) = \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{1} = \boldsymbol{d}}} \langle \boldsymbol{A}, -\boldsymbol{C} \rangle. \quad \text{(A.24)}$$

We recognize this as an optimal transportation problem with cost matrix $-\boldsymbol{C}$, symmetric marginals $\boldsymbol{d}$, with the additional constraint that the diagonal of the coupling must be zero.

3) Proof of Lemmas 5:

Proof. The constrained convex optimization program (A.16) is equivalent to

$$\min_{\substack{\mathrm{diag}(\boldsymbol{A}) = 0 \\ \boldsymbol{1}^T \boldsymbol{A} \boldsymbol{1} = T \\ \boldsymbol{A} = \boldsymbol{A}^T}} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}), \boldsymbol{C} \rangle + \langle \boldsymbol{A}, -\boldsymbol{C} \rangle - \epsilon H_o(\boldsymbol{A}). \quad \text{(A.25)}$$

To find the solution subject to the trace T constraint, we will find the KKT conditions. The Lagrangian is

$$\mathcal{L}(\boldsymbol{A}, \lambda) = \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}), \boldsymbol{C} \rangle + \langle \boldsymbol{A}, -\boldsymbol{C} \rangle - \epsilon H_o(\boldsymbol{A}) + \lambda(\sum_{ij} \boldsymbol{A}_{ij} - 1). \quad \text{(A.26)}$$

Using the symmetry of $\boldsymbol{A}$, the first-order KKT condition is

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{A}_{ij}} \mathcal{L}(\boldsymbol{A}) &= \boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - \boldsymbol{C}_{ij} - \boldsymbol{C}_{ji} \\ &\quad + \epsilon \log(\boldsymbol{A}_{ij}) + \epsilon \log(\boldsymbol{A}_{ji}) - 2\lambda \\ &= 0. \end{aligned} \quad \text{(A.27)}$$

The KKT conditions are therefore

$$\boldsymbol{A}_{ij} = \exp\Big(-\frac{1}{2\epsilon}(\boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - 2\boldsymbol{C}_{ij}) + \lambda \Big), \quad \text{(A.28)}$$

$$\sum_{ij} \boldsymbol{A}_{ij} = T. \quad \text{(A.29)}$$

Therefore, λ is the unique real number number such that $\sum_{ij} \boldsymbol{A}_{ij} = T$, or

$$\sum_{ij} \exp\Big(-\frac{1}{2\epsilon}(\boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - 2\boldsymbol{C}_{ij}) + \lambda \Big) = T. \quad \text{(A.30)}$$

Equivalently, the solution is $T\boldsymbol{A}/\|\boldsymbol{A}\|_1$.

We now proceed with the solution to the linear program (A.13). Examining the linear program, we see that we can rewrite the problem as

$$\min_{\substack{\sum_{i \neq j} \boldsymbol{A}_{ij} = T \\ \boldsymbol{A}_{ij} \geq 0}} \sum_{ij} \boldsymbol{A}_{ij}(\boldsymbol{C}_{ii} + \boldsymbol{C}_{jj} - 2\boldsymbol{C}_{ij}). \quad \text{(A.31)}$$

This linear program has a well-known solution (see, for example, Exercise 4.8 in Boyd et al. [2004a]).

We note that taking $\epsilon \to 0$ in the solution to the entropically regularized LP, we see that $\boldsymbol{A}$ becomes a matrix supported on the entries $ij \in \mathcal{I}$, where $\mathcal{I}$ is defined by (A.19) and all of the a_{ij} are equal. □

4) Proof of Lemma 1: Therefore, the path between these parametrizations is really just the Euclidean path in $\mathbb{S}^n_+$. In particular, this means that if F is convex as a function of $\boldsymbol{L}$, F is convex as a function of $\mathrm{diag}(\boldsymbol{A}\boldsymbol{1}) - \boldsymbol{A}$, or $F \circ \pi$ is convex. Indeed, letting $\boldsymbol{L}(t) = (1-t)\boldsymbol{L} + t\boldsymbol{L}'$ be a path over Laplacians, we see that $\boldsymbol{L}(t) = \pi(\boldsymbol{A}(t))$, where $\boldsymbol{A}(t) = (1-t)\boldsymbol{A} + t\boldsymbol{A}'$. The derivatives also match

$$\begin{aligned} \partial_t F(\boldsymbol{L}(t)) &= \langle \nabla F(\boldsymbol{L}(t)), \boldsymbol{L}'(t) \rangle \\ &= \langle \nabla F(\pi(\boldsymbol{A}(t))), \boldsymbol{L}' - \boldsymbol{L} \rangle \\ &= \langle \nabla F(\pi(\boldsymbol{A}(t))), \mathrm{diag}((\boldsymbol{A}' - \boldsymbol{A})\boldsymbol{1}) - (\boldsymbol{A}' - \boldsymbol{A}) \rangle \\ &= \langle \nabla F(\pi(\boldsymbol{A}(t))), \partial_t \pi(\boldsymbol{A}(t)) \rangle \\ &= \partial_t F(\pi(\boldsymbol{A}(t))). \end{aligned} \quad \text{(A.32)}$$

C. Other Properties of Graph Laplacians

In practice, nodes with large degree may have undue influence on the spectral properties of the graph Laplacian. Therefore, it is useful to also consider normalized versions of the graph Laplacian. In particular, the symmetric normalized graph Laplacian is given by

$$\overline{\boldsymbol{L}} = \boldsymbol{D}^{-1/2}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}. \quad \text{(A.33)}$$

Alternative normalizations include the left and right normalized graph Laplacians, which are $\boldsymbol{D}^{-1}\boldsymbol{L}$ and $\boldsymbol{L}\boldsymbol{D}^{-1}$, respectively.

D. Less Constraints: Variable Trace Linear Programs

In this section, we discuss in more detail what happens to linear programs when we do not constrain the trace or the degree. Suppose we now want to solve the generalization of (A.11) where the trace T is not fixed. In this case, the linear program becomes

$$\min_{\mathrm{diag}(\boldsymbol{A}) = 0,\ \boldsymbol{A}_{ij} \geq 0} \langle \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}), \boldsymbol{C} \rangle + \langle \boldsymbol{A}, -\boldsymbol{C} \rangle. \quad \text{(A.34)}$$

In general, this problem is not well posed. Indeed, writing this linear program in the equivalent form

$$\min_{\boldsymbol{A}_{ij}\geq 0,\ i\neq j} \sum_{ij} \boldsymbol{A}_{ij}(\boldsymbol{C}_{ii}+\boldsymbol{C}_{jj}-2\boldsymbol{C}_{ij}), \tag{A.35}$$

the solution is either 0 or $-\infty$ depending on the signs of the adjoint operator $\mathcal{A}(\boldsymbol{C})$. Furthermore, the entropically regularized program becomes

$$\min_{\mathrm{diag}(\boldsymbol{A})=0} \langle \mathrm{diag}(\boldsymbol{A}\mathbf{1}), \boldsymbol{C}\rangle + \langle \boldsymbol{A}, -\boldsymbol{C}\rangle - \epsilon H_o(\boldsymbol{A}) \tag{A.36}$$

Notice that this is now an unconstrained minimization problem. The solution is given by

$$\boldsymbol{A}_{ij} = \exp\Big(-\frac{1}{2\epsilon}(\boldsymbol{G}_{ii}+\boldsymbol{G}_{jj}-2\boldsymbol{G}_{ij})\Big), \tag{A.37}$$

Again taking $\epsilon \to 0$, we see that all elements ij such that $\boldsymbol{G}_{ii}+\boldsymbol{G}_{jj}-2\boldsymbol{G}_{ij}<0$ blow up, and so the solution to the original LP is again 0 or $-\infty$. We note that the original LP is equivalent to

$$\min_{T,\boldsymbol{L}\in\mathcal{L}_n(T)} \langle \boldsymbol{L}, \boldsymbol{C}\rangle. \tag{A.38}$$

We could consider adding a Laplacian trace regularization term, which would yield the augmented linear program

$$\min_{\mathrm{diag}(\boldsymbol{A})=0,\ \boldsymbol{A}_{ij}\geq 0} \langle \mathrm{diag}(\boldsymbol{A}\mathbf{1}), \boldsymbol{C}\rangle + \langle \boldsymbol{A}, -\boldsymbol{C}\rangle - \lambda \sum_{ij} \boldsymbol{A}_{ij}. \tag{A.39}$$

and the entropically regularized program

$$\min_{\mathrm{diag}(\boldsymbol{A})=0} \langle \mathrm{diag}(\boldsymbol{A}\mathbf{1}), \boldsymbol{C}\rangle + \langle \boldsymbol{A}, -\boldsymbol{C}\rangle - \epsilon H_o(\boldsymbol{A}) - \lambda \sum_{ij} \boldsymbol{A}_{ij}. \tag{A.40}$$

The entropically regularized program has solution

$$\boldsymbol{A}_{ij} = \exp\Big(-\frac{1}{2\epsilon}(\boldsymbol{C}_{ii}+\boldsymbol{C}_{jj}-2\boldsymbol{C}_{ij})+\lambda\Big), \tag{A.41}$$

E. Fixed Degree and Sinkhorn

As we mention previously, suppose that we fix the degree of the nodes in the vector $\boldsymbol{d}$. In this case, we must solve the linear program

$$\min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0,\ \mathrm{diag}(\boldsymbol{A})=\mathbf{0} \\ \boldsymbol{A}\mathbf{1}=\boldsymbol{d}}} \langle \boldsymbol{A}, -\boldsymbol{C}\rangle. \tag{A.42}$$

To enforce $\mathrm{diag}(\boldsymbol{A})=0$, we can set the diagonal of $\boldsymbol{C}$ to be $-\infty$. Furthermore, if we know the sparsity pattern of $\boldsymbol{A}$, then we can set the corresponding entries of $\boldsymbol{C}$ to be $-\infty$ as well.

Let us consider the most constrained case then of fixed degree $\boldsymbol{d}$ and fixed sparsity pattern given by an edge set E.

$$\min_{\substack{\boldsymbol{A}\in\mathcal{H}_n \\ \boldsymbol{A}\geq 0, \\ \boldsymbol{A}\mathbf{1}=\boldsymbol{d}}} \langle \boldsymbol{A}, \boldsymbol{C}'\rangle. \tag{A.43}$$

where

$$\boldsymbol{C}'_{ij} = \begin{cases} \infty, & i=j \text{ or } ij\notin E, \\ -C_{ij}, & \text{else.} \end{cases} \tag{A.44}$$

We could try to approximately solve this using Sinkhorn's algorithm.

Alternatively, in the fixed degree case, it is straightforward to show that the entropic mirror descent updates take the form of an exponential reweighting step followed by a Sinkhorn projection.

F. Extra Figures

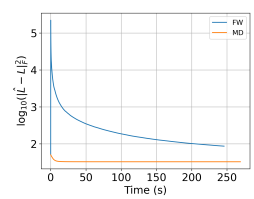

Fig. 6. Convergence for projection of a noisy 1000×1000 Laplacian. The convex solver in CVXPY failed to converge on a personal machine.