Birkhoff's Decomposition Revisited
Sparse Scheduling for High-Speed Circuit Switches

Valls, Víctor ; Tassiulas, Leandros ; Iosifidis, George

DOI
10.1109/TNET.2021.3088327

Publication date
2021

Document Version
Accepted author manuscript

Published in
IEEE/ACM Transactions on Networking

Citation (APA)
Valls, V., Tassiulas, L., & Iosifidis, G. (2021). Birkhoff's Decomposition Revisited: Sparse Scheduling for High-Speed Circuit Switches. *IEEE/ACM Transactions on Networking*, *29*(6), 2399-2412. Article 9509349. https://doi.org/10.1109/TNET.2021.3088327

Birkhoff's Decomposition Revisited: Sparse Scheduling for High-Speed Circuit Switches

Víctor Valls, George Iosifidis, and Leandros Tassiulas, *Fellow, IEEE*

***Abstract*— Data centers are increasingly using high-speed circuit switches to cope with the growing demand and reduce operational costs. One of the fundamental tasks of circuit switches is to compute a sparse collection of switching configurations to support a traffic demand matrix. Such a problem has been addressed in the literature with variations of the approach proposed by Birkhoff in 1946 to decompose a doubly stochastic matrix exactly. However, the existing methods are heuristic and do not have theoretical guarantees on how well a collection of switching configurations (i.e., permutations) can approximate a traffic matrix (i.e., a scaled doubly stochastic matrix). In this paper, we revisit Birkhoff's approach and make three contributions. First, we establish the first theoretical bound on the sparsity of Birkhoff's algorithm (i.e., the number of switching configurations necessary to approximate a traffic matrix). In particular, we show that by using a subset of the admissible permutation matrices, Birkhoff's algorithm obtains an ϵ-approximate decomposition with at most $O(\log(1/\epsilon))$ permutations. Second, we propose a new algorithm, `Birkhoff+`, which combines the wealth of Frank-Wolfe with Birkhoff's approach to obtain sparse decompositions in a fast manner. And third, we evaluate the performance of the proposed algorithm numerically and study how this affects the performance of a circuit switch. Our results show that `Birkhoff+` is superior to previous algorithms in terms of throughput, running time, and number of switching configurations.**

***Index Terms*— Scheduling algorithms, machine learning algorithms, switches, Birkhoff decomposition.**

I. Introduction

DATA centers are increasingly adopting hybrid switching designs that combine traditional electronic packet switches with high-speed circuit switches [1]–[3]. In short, packet switches are flexible at making forwarding decisions at a packet level, but have limited capacity and are becoming increasingly expensive in terms of cost, heat, and power. In contrast, circuit switches provide significantly higher data rates at a lower cost but are less flexible at making forwarding decisions. The main drawback of circuit switches is that they have high reconfiguration times, which limit the amount of traffic they can carry [1], [4], [5]. For instance, circuit switches have reconfiguration times in the order of milliseconds (e.g., 25 ms for off-the-self circuit switches [6], [7]), whereas the reconfiguration times in electronic switches are in the scale of microseconds. As a result, hybrid switching architectures load balance and use circuit switches for high-intensity/bursty flows [8], [9][1] and electronic switches for traffic that needs of a more fine-grained scheduling (e.g., delay-sensitive applications).

The problem of computing switching configurations for circuit switches is central to networking and has a direct impact on the performance of nowadays data centers. Mathematically, we can model a circuit switch as a crossbar,[2] and cast the problem of finding a small collection of switching configurations as decomposing a doubly stochastic matrix[3] as a *sparse* convex combination of permutations matrices. In brief, for a given $n \times n$ doubly stochastic matrix $X^\star$ (i.e., a scaled traffic matrix) and an $\epsilon \geq 0$, the goal is to find a *small* collection of permutation matrices $P_1, P_2, \ldots, P_k$ (i.e., switching configurations) and weights $\theta_1, \theta_2, \ldots, \theta_k > 0$ (i.e., the fraction of time the switching configurations will be used) with $\sum_{i=1}^{k} \theta_i \leq 1$ such that

$$\left\| X^\star - \sum_{i=1}^{k} \theta_i P_i \right\|_F \leq \epsilon, \tag{1}$$

where $\| \cdot \|_F$ is the Frobenius norm (see definition in Section III-A). The smaller ϵ is, the higher the throughput. However, practical systems cannot use as many switching configurations as desired as each inflicts a reconfiguration time δ that affects the fraction of time the switch can carry traffic.[4] Or put differently, there is a constraint on the number of configurations a switch can use to approximate a traffic matrix.

Previous work has addressed the problem above with variations (e.g., [8], [14]–[16]) of the approach proposed by Birkhoff in 1946 [17] to decompose a doubly stochastic matrix

Manuscript received October 24, 2020; revised March 25, 2021; accepted April 3, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor I.-H. Hou. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie under Agreement 795244, in part by the European Union Horizon 2020 Research and Innovation Program [Network intelligence for aDAptive and sElf-Learning MObile Networks (DAEMON)] under Agreement 101017109, in part by the NSF under Award 1815676, in part by Army Research Office (ARO) under Grant W911NF1810378, in part by the National Science Foundation (NSF) under Grant CNS-1955204, and in part by the NSF Engineering Research Center (ERC) on Quantum Networks under Grant 581207. *(Corresponding author: Víctor Valls.)*

Víctor Valls and Leandros Tassiulas are with the Electrical Engineering and Institute for Network Science, Yale University, New Haven, CT 06520 USA (e-mail: victor.valls@yale.edu).

George Iosifidis is with the Department of Software Technology, TU Delft, 2628 Delft, The Netherlands.

Digital Object Identifier 10.1109/TNET.2021.3088327

[1] Traffic in data centers is often bursty [10], [11] and uses few input/output ports [12].

[2] See, for example, [13, Section 4.1].

[3] A matrix is doubly stochastic if its entries are non-negative and the sum of every row and column is equal to one. A permutation matrix is a binary doubly stochastic matrix.

[4] Technically, a traffic matrix $X^\star$ is valid for a time window period W, and the decomposition must satisfy $\sum_{i=1}^{k}(\theta_i + \delta) \leq W$. That is, the time spent transmitting $(\sum_{i=1}^{k} \theta_i)$ and reconfiguring (δk) cannot exceed the time window duration (W).

TABLE I

SPARSITY AND PERMUTATION SELECTION COMPLEXITY OF Birkhoff+ (THIS PAPER) AND PREVIOUS ALGORITHMS. LP AND QP STAND FOR LINEAR AND QUADRATIC PROGRAM RESPECTIVELY

Algorithm	Sparsity (k)	Perm. selec. complexity
Birkhoff [17]	—	One LP
Solstice [8]	—	Multiple LPs
Eclipse* [9]	Approx. ratio	Multiple LPs
FW [18]	$O(1/\epsilon^2)$	One LP
FCFW [19]	$O(\log(1/\epsilon))$	One LP + QP(k)
Birkhoff+ (this paper)	$O(\log(1/\epsilon))$	One LP

exactly (i.e., $\epsilon = 0$). However, little is known about the behavior or convergence properties of Birkhoff's algorithm, and so fundamental questions remain still unanswered. In particular, how does the decomposition approximation ϵ in Eq. (1) depend on the number of switching configurations? How much does an additional switching configuration contribute to increasing a circuit switch throughput? How is Birkhoff's algorithm related to other numerical methods in other fields, such as optimization and machine learning? Answering these questions is crucial to better understand the structure of the problem and to design new algorithms that improve the performance of circuit switches. To this end, the main contributions of the paper are the following:

(i) Revisiting Birkhoff's Approach: We revisit Birkhoff's algorithm and establish the first theoretical bound on its sparsity (i.e., the number of permutations necessary to approximate a doubly stochastic matrix). In particular, we show that by selecting permutations from a subset of admissible permutations, Birkhoff's algorithm has sparsity $O(\log(1/\epsilon))$ (Theorem 1). That is, the number of permutations required to obtain an ϵ-approximate decomposition increases logarithmically with the decomposition error. Our results also show that previous Birkhoff-based algorithms that select permutations using a Max-Min criterion (e.g., [8]) have logarithmic sparsity (Corollary 1), and that Birkhoff's algorithm is strongly connected to block-coordinate descent and Frank-Wolfe methods in convex optimization (Section IV-D.2 and Section V).

(ii) New Algorithm (Birkhoff+): We propose a new algorithm that combines Birkhoff's approach and Frank-Wolfe. Specifically, permutation matrices are selected using a Frank-Wolfe-type update with a barrier function, while the weights as in Birkhoff's approach. The proposed algorithm has theoretical guarantees (Corollary 2) and is non-trivial as a direct combination of Birkhoff's approach with Frank-Wolfe may not converge (Theorem 4). Furthermore, Birkhoff+ is faster than previous algorithms as it computes a new permutation/configuration by solving a *single* linear program (LP). Table I contains a summary of the main differences between Birkhoff+ and the state-of-the-art algorithms discussed in Section II.

(iii) Numerical Evaluation: We evaluate Birkhoff+'s performance in a circuit switch application and compare it against existing algorithms for a range of matrices (dense, sparse, skewed) that capture the characteristics of traffic in data centers. Our results show that Birkhoff+ is superior to previous algorithms in terms of throughput, running time, and number of switching configurations. For instance, when $\delta/W = 10^{-2}$ (the reconfiguration time over the time available for transmission), Birkhoff+ has 7% more throughput than the best state-of-the-art algorithm. If we consider, in addition, the time to compute the switching configurations as an overhead, the throughput gain increases to 34% (switch with $n = 100$ ports).

The outline of the paper is as follows. Section II presents related work and Section III the preliminaries, which include the notation and how to find a permutation matrix by solving a linear program. In Section IV, we revisit Birkhoff's approach in a *general* form, establish its sparsity rate, and show how this is connected to block-coordinate descent methods in convex optimization. The latter also clarifies that selecting a permutation matrix can be seen as choosing a (gradient) descent direction. Section V shows how to use Frank-Wolfe algorithms to decompose a doubly stochastic matrix, and how Frank-Wolfe chooses a permutation matrix that provides "steepest descent." In Section VI, we present the new algorithm (Birkhoff+) and in Section VII evaluate its performance against the state-of-the-art algorithms. Finally, Section VIII concludes. All the proofs are in the Appendix.

II. HISTORY AND RELATED WORK

A. Birkhoff's Approach

This is the method employed by Birkhoff in 1946 to decompose a doubly stochastic matrix *exactly* [17, first theorem].[5] In brief, the method consists of finding permutations matrices sequentially (e.g., with the Hungarian algorithm) and terminates when it obtains an exact decomposition, which happens with at most $k = (n-1)^2 + 1$ iterations/permutations [21], [22] by Carathéodory's theorem. The method, however, does not guarantee that the decomposition (i.e., $\sum_{i=1}^{k} \theta_i P_i$) is close to the doubly stochastic matrix it aims to approximate (i.e., $X^\star$). In fact, the approximation is typically very poor until the algorithm converges exactly in the last iteration (see Figure 3a in Section VII for an example).

B. Related Mathematical Problems

The problem of finding the Birkhoff decomposition with the minimum number of permutation matrices ($\min k$ s.t. $X^\star = \sum_{i=1}^{k} \theta_i P_i$) was addressed in [15] and shown to be NP-hard. In [23], the authors also show that the problem is not tractable when the minimal decomposition can be expressed with $k \geq 4$ permutations. The work in [24] formulates a similar problem. For a demand matrix $D = X^\star - S$ with $S \in [0,1]^{n\times n}$,[6] the goal is to find a collection of weights $\{\theta_i\}_{i=1}^{k}$ and permutation matrices $\{P_i\}_{i=1}^{k}$ that minimizes $\sum_{i=1}^{k}(\theta_i + \delta)$ subject to $\sum_{i=1}^{k} \theta_i P_i \geq D$ entry-wise. The problem is shown to be NP-complete. The problem addressed in this paper is different

[5]The result is also known as Birkhoff-von Neumman (BvN) as it was discovered independently by von Neumman [20]. We use Birkhoff instead of BvN as the algorithm used in the literature is based on the method of proof used by Birkhoff in [17].

[6]The entries of the demand matrix D are non-negative. Matrix S adds a non-negative virtual load to demand matrix so that $D+S$ is doubly stochastic.

in spirit from the mathematical problems in [15], [24] because we do not aim to find a (small) collection of objects (i.e., k) subject to decomposition constraints (i.e., $X^\star = \sum_{i=1}^k \theta_i P_i$ or $\sum_{i=1}^k \theta_i P_i \geq D$). Instead, our goal is to design an algorithm that minimizes $\|X^\star - X_k\|_F$ where $X_k = \sum_{i=1}^k \theta_i P_i$. The convergence rate of the numerical method corresponds to the number of permutations required to obtain an ϵ-approximate decomposition.

C. Algorithms

The paper in [8] proposes `Solstice`, a Birkhoff-based heuristic for finding a Birkhoff decomposition with few permutations/configurations. `Solstice` picks permutation matrices using a Max-Min type criterion, and the weights or configurations durations are selected as large as possible provided $X^\star - \sum_{i=1}^k \theta_i P_i$ is non-negative entry-wise. The work in [9] proposes `Eclipse`, a sub-modular-type algorithm for solving the problem of the type introduced in [24]. Permutation matrices and weights are selected jointly to maximize an effective utilization criterion, which takes into account the reconfiguration penalty δ. Also, [9] shows that the final decomposition satisfies the optimal approximation ratio in sub-modular optimization with cover constraints. Both algorithms [8], [9] select permutation matrices by solving multiple linear programs (LPs) with a simplex type method [25]. Finally, we note the recent works in [16] and [26]. The first extends `Eclipse` to use a special type of weights/time coefficients that do not constraint the decomposition to be a scaled doubly stochastic matrix. The second addresses the *online* version of the problem in [9]—in the machine learning sense [27]—where the traffic matrix is learned a posteriori.

To conclude, we note the Frank-Wolfe algorithms [18], [28] used extensively in machine learning. The Frank-Wolfe setup is the following. Given a collection of discrete objects $\mathcal{D}$ and a convex set $\mathcal{X} \subseteq \text{conv}(\mathcal{D})$, the goal is to minimize a convex function by making convex combinations of the discrete objects. The problem addressed in this paper can be seen as a special case for Frank-Wolfe. The permutation matrices correspond to the discrete objects, the Birkhoff polytope is the convex set, and the objective function a metric that captures the distance between the approximate decomposition and $X^\star$ (e.g., Frobenius norm or Euclidean distance). Also, unlike Birkhoff-based approaches, Frank-Wolfe algorithms provide sparsity guarantees and ensure that the approximate decomposition is always a doubly stochastic matrix.

III. Preliminaries

A. Notation

We use $\mathbf{R}_+$ and $\mathbf{R}^d$ to denote the set of nonnegative real numbers and d-dimensional real vectors. Vectors and matrices are written in lower and upper case respectively, and all vectors are in column form. The transpose of a vector $x \in \mathbf{R}^d$ is indicated with x^T, and we use $\mathbf{1}$ to indicate the all ones vector—the dimension of the vector will be clear from the context. We use parenthesis to indicate an element in a vector, i.e., $x(j)$ is the j'th element of vector x. Similarly, the element in the i'th row and j'th column of a matrix X is indicated with $X(i,j)$. For two vectors $x, y \in \mathbf{R}^d$, we write $x \succ y$ when $x(j) > y(j)$ for all $j \in \{1, \ldots, d\}$, and $x \succeq y$ when $x(j) \geq y(j)$. Finally, we recall the Frobenius norm of a matrix X is defined as $\|X\|_F = \sqrt{\sum_{i,j} |X(i,j)|^2} = \sqrt{\text{Tr}(XX^*)}$ and that $[n]$ is the short-hand notation for $\{1, \ldots, n\}$.

Algorithm 1 `General Birkhoff`

Input: Doubly stochastic matrix $X^\star$, $\epsilon \geq 0$, and $k_{\max} \geq 1$
Set: $k = 1$ and $X_0 = \{0\}^{n \times n}$
while $\|X_{k-1} - X^\star\|_F > \epsilon$ and $k \leq k_{\max}$ **do**
- Compute P_k, θ_k that satisfy Eqs. (3)-(5)

$X_k \leftarrow X_{k-1} + \theta_k P_k$
$k \leftarrow k + 1$
end while
return $(P_1, \ldots, P_{k-1})$, $(\theta_1, \ldots, \theta_{k-1})$

B. Finding Extreme Points by Solving Linear Programs

We will present algorithms that find extreme points (i.e., permutation matrices) by solving linear programs (LPs) over a convex set (i.e., the Birkhoff polytope or set of doubly stochastic matrices). We recall the following result from linear programming.

Lemma 1: Let $\mathcal{X}$ be a bounded polytope from $\mathbf{R}^d$, and $\mathcal{E}$ denote its extreme points. For any vector $c \in \mathbf{R}^d$, we have that $\{\arg\min_{x \in \mathcal{X}} \ c^T x\} \cap \mathcal{E} \neq \emptyset$.

That is, an extreme point in $\mathcal{E}$ is always a solution to $\min_{x \in \mathcal{X}} c^T x$. In our case, $\mathcal{X}$ is the Birkhoff polytope and $\mathcal{P}$ the set of permutation matrices. Throughout the paper, we will cast linear programs as

$$\texttt{LP}(c, \mathcal{X}): \quad \begin{array}{l} \text{minimize } c^T x \\ \text{subject to } x \in \mathcal{X}, \end{array} \tag{2}$$

and we will assume that the solution returned is always an extreme point—which is the case if we solve the LP with a simplex-type method [25].

IV. Revisiting Birkhoff's Algorithm

This section revisits Birkhoff's algorithm. The main technical contribution is Theorem 1, which establishes that the number of permutation matrices in Birkhoff's approach increases logarithmically with the decomposition error.

A. Approximate Birkhoff Decomposition Problem

The mathematical problem we want to solve is the following. For a given $n \times n$ doubly stochastic matrix $X^\star$ and an $\epsilon \geq 0$, our goal is to find a *small* collection of permutation matrices $P_1, P_2, \ldots, P_k$ and weights $\theta_1, \theta_2, \ldots, \theta_k > 0$ with $\sum_{i=1}^k \theta_i \leq 1$ such that $\|X^\star - \sum_{i=1}^k \theta_i P_i\|_F \leq \epsilon$. Recall a matrix $X \in [0,1]^{n \times n}$ is doubly stochastic if every row and column sums to one. That is, $X\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^T X = \mathbf{1}^T$. Also, a doubly stochastic matrix is a permutation if its entries are binary.

B. Algorithm Description

The original Birkhoff algorithm is described in Algorithm 1, and consists of two steps. First, the algorithm computes a permutation P_k and a weight $\theta_k > 0$ that satisfy the following three conditions:[7]

$$X_{k-1}(a,b) + \theta_k P_k(a,b) \leq X^\star(a,b) \qquad \forall a,b \in [n] \quad (3)$$

$$\theta_k > 0 \qquad \forall k \geq 1 \quad (4)$$

$$\textstyle\sum_{i=1}^{k} \theta_i \leq 1 \qquad \forall k \geq 1 \quad (5)$$

The second step is to add $\theta_k P_k$ to the previous approximate decomposition, i.e., $X_k = X_{k-1} + \theta_k P_k$. Hence, $X_k(a,b) \leq X^\star(a,b)$ for all $a,b \in \{1,\dots,n\}$. The algorithm terminates when the approximate decomposition X_k is ϵ close to $X^\star$, or when the maximum number of admissible permutations ($k_{\max}$) is reached. See [29, Fig. 2] for a decomposition example.

C. Convergence

We proceed to establish the convergence of Birkhoff's algorithm. We start by presenting the following lemma, which lower and upper bounds $\|X_k - X^\star\|_F$.

Lemma 2: Consider the setup in Algorithm 1. The following two bounds hold:

$$\|X_k - X^\star\|_F \geq \left(1 - \sum_{i=1}^{k} \theta_i\right) \quad (6)$$

$$\|X_k - X^\star\|_F \leq \sqrt{n \prod_{i=1}^{k} \left(1 - \frac{n\theta_i^2}{\|X_{i-1} - X^\star\|_F^2}\right)} \quad (7)$$

where $\theta_i \leq \frac{1}{\sqrt{n}} \|X_{i-1} - X^\star\|_F$.

Proof: See the Appendix. ∎

The bounds in Lemma 2 are very general as they hold for any collection of permutation matrices and weights that satisfy the conditions in Eqs. (3)-(5). The lower bound in Eq. (6) tells us that the approximate decomposition error is at least $(1 - \sum_{i=1}^{k} \theta_i)$, and so we will have an exact decomposition (i.e., serve 100% of the traffic demand) only if $\sum_{i=1}^{k} \theta_i = 1$. The upper bound in Eq. (7) shows how the decomposition error depends on the weights θ_i and the previous approximations $\|X_{i-1} - X^\star\|_F^2$, $i = 1,\dots,k$. In particular, on the ratio $n\theta_i^2 / \|X_{i-1} - X^\star\|_F^2$, which captures how large θ_i is with respect to the previous approximation. Note that the values that θ_i can take depend on $\|X_{i-1} - X^\star\|_F^2$ as we must always satisfy the conditions in Eqs. (3)–(5). Finally, we note that finding a joint collection of weights and permutation matrices that minimize the RHS of Eq. (7) for a fixed k is as difficult as minimizing $\|X_k - X^\star\|_F^2$ directly, since the RHS of Eq. (7) depends on $\|X_{i-1} - X^\star\|_F$, $i = 1,\dots,k$. Because of the latter, we study how to minimize the RHS of Eq. (7) in an iterative manner: for a given collection of permutation matrices P_i and weights θ_i with $i = 1,\dots,k-1$, our goal is to find a permutation matrix P_k and weight θ_k that decrease the RHS of Eq. (7).

[7]The procedure proposed by Birkhoff in 1946 [17] does not specify how to compute such permutation and weight. The subroutine PERM described in Algorithm 2—which we will present in Theorem 1—returns a permutation P_k and a weight θ_k that satisfy Eqs. (3)-(5).

Algorithm 2 Subroutine PERM

1: **Input:** $X^\star$ and $X_{k-1} = \sum_{i=1}^{k-1} \theta_i P_i$
2: $\alpha \leftarrow (1 - \sum_{i=1}^{k-1} \theta_i)/n^2$
3: $P_k \leftarrow \hat{P} \in \mathcal{I}_k(\alpha)$
4: $\theta_k \leftarrow$ BIRKHOFF_STEP$(X^\star, X_{k-1}, P_k)$ (Algorithm 3)

In the following, we study the algorithm's progress in terms of error for every additional permutation matrix in the decomposition. Addressing this question is important to bound the number of permutations required to obtain an ϵ-approximate decomposition and to know how to select "good" permutation matrices. To start, let $\mu_i := n\theta_i^2 / \|X_{i-1} - X^\star\|_F^2$ and rewrite Eq. (7) as

$$\|X_k - X^\star\|_F \leq \sqrt{n \prod_{i=1}^{k} (1 - \mu_i)} \quad (8)$$

Note that $(1 - \mu_i) \in [0,1)$ for all $i = 1,2,\dots$ since $\theta_i \geq 0$ and $\theta_i \leq \frac{1}{\sqrt{n}} \|X_{i-1} - X^\star\|_F$ by Lemma 2. Hence, we have that $X_k \to X^\star$ as $k \to \infty$ and so the algorithm converges. Now, suppose there exists a constant $\mu_{\min} > 0$ such that $\mu_{\min} \leq \mu_i$ for all $i \geq 1$. Then, the bound in Eq. (8) simplifies to

$$\|X_k - X^\star\|_F \leq \sqrt{n}(1 - \mu_{\min})^{k/2}. \quad (9)$$

The last equation tells us that the approximation error decreases exponentially with the number of permutations. For example, if $\mu_{\min} = 1/2$, we have that $\|X_k - X^\star\|_F \leq \sqrt{n}(1/2)^{k/2}$, which means that every additional permutation in the decomposition decreases the approximation error by at least half. The ratio $\kappa := 1/\mu_{\min} \geq 1$ can be regarded as the condition number in convex optimization with a smooth and strongly convex objective function [30][Section 9.1.2 and 9.3.1].

The following lemma establishes an upper bound on the number of permutations required to obtain an ϵ-approximate decomposition provided that a constant $\mu_{\min} > 0$ exists.

Lemma 3: Suppose $n\theta_i^2 / \|X_{i-1} - X^\star\|_F^2 \geq \mu_{\min}$ *for all* $i = 1,\dots,k$ *for some constant* $\mu_{\min} > 0$. *Then, Algorithm 1 obtains an* ϵ*-approximate decomposition with at most*

$$k \leq 2 \log^{-1}\left(\frac{1}{1 - \mu_{\min}}\right) \log\left(\frac{\sqrt{n}}{\epsilon}\right)$$

permutation matrices.

Lemma 3 says that if a constant $\mu_{\min}$ exists, then the number of permutations required to obtain an ϵ-approximate decomposition has a logarithmic dependence with ϵ. Hence, it remains to show whether such constant exists. Or equivalently, that we can select a θ_i such that $n\theta_i^2 / \|X_{i-1} - X^\star\|_F^2$ is uniformly lower bounded by a strictly positive constant. We show that in the following theorem, which is one of the main contributions of the paper.

Algorithm 3 BIRKHOFF_STEP

1: **Input:** $X^\star$, X_{k-1}, and P_k
2: **return** $\min_{a,b}\{(X^\star(a,b) - X_{k-1}(a,b) - 1)P_k(a,b) + 1\}$

Theorem 1: Consider Algorithm 1 and replace step • with the subroutine PERM (Algorithm 2) where

$$\mathcal{I}_k(\alpha) = \{P \in \mathcal{P} \mid X_{k-1}(a,b) + \alpha P(a,b) \leq X^\star(a,b)\} \tag{10}$$

with $\alpha = \sqrt{\frac{\mu_{\min}}{n}}(1 - \sum_{i=1}^{k}\theta_i)$ *and* $\mu_{\min} = 1/n^3$. *Algorithm 1 obtains an* ϵ*-approximate decomposition with at most*

$$k \leq 2\log^{-1}\left(1 - \min_{i\in[k]} \frac{n\theta_i^2}{\|X_{i-1} - X^\star\|_F^2}\right)^{-1} \log\left(\frac{\sqrt{n}}{\epsilon}\right) \tag{11}$$

permutation matrices.

Proof: See the Appendix. ∎

Theorem 1 establishes that by selecting permutation matrices from set $\mathcal{I}_k(\alpha) \subseteq \mathcal{P}$, and weights as indicated in Algorithm 2, then the number of permutation matrices required to obtain an ϵ-decomposition increases logarithmically with ϵ. Set $\mathcal{I}_k(\alpha)$ is necessary to enforce that the conditions in Eqs. (3)–(5) are satisfied, but also to push Birkhoff's algorithm to make sufficient progress in every iteration. Observe that the threshold α is bounded away from zero and that this depends on the constant $\mu_{\min} = 1/n^3$. Finally, we have written $\min_{i\in\{1,\ldots,k\}} n\theta_i^2/\|X_{i-1} - X^\star\|_F^2$ instead of $\mu_{\min}$ in Eq. (11) (c.f. Lemma 3) to emphasize two points. The first one is that $\mu_{\min}$ is over-conservatively small, and that we can in general obtain a much sharper upper bound. In the numerical evaluation (Section VII-B.1), we show the condition numbers ($\kappa = 1/\mu_{\min}$) of different algorithms. The second point is that $n\theta_i^2/\|X_{i-1} - X^\star\|_F^2$ is a quantity that we can measure and so use as a criterion for selecting a "good enough" permutation matrix. Importantly, note that the PERM subroutine does not specify which specific permutation to select from set $\mathcal{I}_k(\alpha)$, which is in marked contrast to previous approaches (e.g., [8], [9], [16]), which use a predefined criterion for selecting permutation matrices and weights.

D. Discussion

1) Max-Min Birkhoff Algorithms: The most popular variant of Birkhoff's algorithm (e.g., [8], [15]) aims to find a permutation matrix with the largest associated weight. Such approach corresponds to solving the following optimization problem:

$$\begin{aligned} \underset{\theta>0, P\in\mathcal{P}}{\text{maximize}} \quad & \theta \\ \text{subject to} \quad & X_{k-1}(a,b) + \theta P(a,b) \leq X^\star(a,b) \\ & \forall a,b \in [n] \end{aligned} \tag{12}$$

The strategy is also known as Max-Min because it is equivalent to finding a permutation matrix P with the largest smallest element $X^\star(a,b) - X_k(a,b)$ provided $P(a,b) = 1$. Hence, the set of solutions to the optimization in (12) is a subset of $\mathcal{I}_k(\alpha)$ since this includes all the solutions with $\theta \geq \alpha > 0$. Further, since $\mathcal{I}_k(\alpha) \neq \emptyset$ by Theorem 1, the set of solutions of problem (12) is also non-empty. We have arrived at the following corollary to Theorem 1.

Corollary 1 (Theorem 1): The Birkhoff-type algorithms that select permutation matrices using a Max-Min criterion (e.g., [8]) have sparsity $O(\log(1/\epsilon))$.

To conclude, we would like to emphasize that solving the problem in (12) is non-trivial. The typical approach is to fix a weight θ, and then try to find a permutation matrix that satisfies the constraint $X_{k-1}(a,b) + \theta P(a,b) \leq X^\star(a,b)$ for all $a,b \in [n]$. The process is repeated for different weights, which are selected with different strategies; for example, [8] uses a halving threshold rule. The main issue with this method is that it is slow, and so it does not suit applications that need to carry out decomposition fast. For example, when we are given a traffic matrix associated with a time window. The time spent computing the switching configurations is time that the switch cannot use for serving traffic.

2) Birkhoff's Algorithm as a Block-Coordinate Descent: The Birkhoff algorithm can be thought in convex optimization terms. In particular, as solving the following convex optimization problem

$$\begin{aligned} \underset{X\in\mathbf{R}^{n\times n}}{\text{minimize}} \quad & \|X - X^\star\|_F^2 \\ \text{subject to} \quad & X(a,b) \leq X^\star(a,b) \qquad \forall a,b \in [n] \\ & X(a,b) \geq 0 \qquad\qquad\quad \forall a,b \in [n] \end{aligned} \tag{13}$$

using a block-coordinate descent method with $X_0 = \{0\}^{n\times n}$ (see [31]–[33][Section 7.5.3]). Note that the objective is convex and the constraints linear. The block-coordinate method consists of the update[8]

$$X_k = X_{k-1} + \theta_k M_k$$

where $\theta_k > 0$ is a step size and $M_k \in \{-1, 0, 1\}^{n\times n}$ a matrix that indicates the direction in which to update each of the coordinates. Birkhoff's approach can be regarded as a special case where the M_k matrices are permutations, and so have constrains on the group of coordinates can be *jointly* updated. Also, there are no negative coordinates since by selecting $X_0 = \{0\}^{n\times n}$ as starting point the algorithm only needs to "move forward." To conclude, we note that our sparsity result is connected to the linear convergence rate obtained by convex optimization algorithms that exploit the smoothness and strong convexity of the objective function [30, Ch. 4].

V. Frank-Wolfe for the Approximate Birkhoff Decomposition

In this section, we show how the Frank-Wolfe (FW) algorithm and its fully corrective variant (FCFW) can be used to decompose a doubly stochastic matrix. The main contributions are to give explicit sparsity bounds for the FW and FCFW algorithms (Theorem 2 and 3) and to discuss their drawbacks (low sparsity and high complexity, respectively). We also compare how Frank-Wolfe and Birkhoff select permutation

[8]The method is usually expressed in vector form. In our case, we can create a vector by stacking the matrix columns.

Algorithm 4 `Birkhoff (Vector Form)`

Input: Birkhoff polytope $\mathcal{B}$, $x^\star \in \mathcal{B}$, $\epsilon \geq 0$, $k_{\max} \geq 1$
Set: $k = 1$ and $x_0 = 0$
while $\|x^\star - x_{k-1}\|_2 > \epsilon$ and $k \leq k_{\max}$ **do**
  $\circ$ $p_k \leftarrow$ `LP` $(-\lceil x^\star - x_{k-1} \rceil, \mathcal{B})$
  $\star$ $\theta_k \leftarrow$ `BIRKHOFF_STEP`$(x^\star, x_{k-1}, p_k)$
  $\diamond$ $x_k \leftarrow x_{k-1} + \theta_k p_k$
    $k \leftarrow k + 1$
end while
return $(p_1, \ldots, p_{k-1})$, $(\theta_1, \ldots, \theta_{k-1})$

matrices (Observation 1) and show that Frank-Wolfe provides the "steepest descent" permutation (Observation 2). The latter will be key for selecting permutation matrices in the `Birkhoff+` algorithm we will present in Section VI.

A. Birkhoff Polytope and Algorithm in Vector Form

In the rest of the paper, it will be more convenient to write $n \times n$ doubly stochastic matrices as n^2-dimensional vectors[9] in the set

$$\mathcal{B} := \{x \in \mathbf{R}^d \mid x \succeq 0,\ Ax = b\},$$

where $d = n^2$, $A \in \{0,1\}^{2n \times d}$, and $b := \{1\}^{2n}$. Matrix A contains the $2n$ equality constraints that characterize the Birkhoff polytope (i.e., the sum of the columns and rows of a doubly stochastic matrix must be equal to 1). The specific structure of A can be derived easily and is given in the Appendix. As before, we use set $\mathcal{P} \subset \{0,1\}^d$ to denote the set of permutation matrices or extreme points, but now these are in column form. The terms extreme point and permutation matrix will be used interchangeably in the rest of the paper. Finally, Algorithm 4 contains the procedure of the classic Birkhoff algorithm [17] in vector form,[10] which is a special case of the more general Algorithm 1. Permutation matrices are selected by solving the linear program `LP` $(-\lceil x^\star - x_{k-1} \rceil, \mathcal{B})$ and the step sizes as large as possible provided $x_k \preceq x^\star$ for all $k \geq 1$ (see Section III-B).[11] The `LP` $(-\lceil x^\star - x_{k-1} \rceil, \mathcal{B})$ returns any admissible permutation matrix (see Section III) and $\lceil \cdot \rceil$ denotes the entry-wise ceiling of a vector.

B. Frank-Wolfe Overview

In short, the Frank-Wolfe algorithm is a numerical method for minimizing a convex function f over a convex set contained in the convex hull of a set of discrete points or atoms [28]. In our case, the convex set is the Birkhoff polytope ($\mathcal{B}$) and the atoms the set of permutation matrices ($\mathcal{P}$). In each iteration, the algorithm selects an extreme point with the update

$$p_k \in \arg\min_{u \in \mathcal{P}} \ \nabla f(x_{k-1})^T u \tag{14}$$

[9]Instead of having a matrix $Z \in \mathbf{R}_+^{n \times n}$ such that $Z\mathbf{1} = Z^T\mathbf{1} = \mathbf{1}$, we work with a vector $x := (z_1, \ldots, z_n)$ where z_i is the i'th column of Z.

[10]The algorithm corresponds to the method of proof employed by Birkhoff to show that a doubly stochastic matrix is an arithmetic measure of permutation matrices. See [17], theorem on page 1.

[11]This permutation choice satisfies Eqs. (3)-(5).

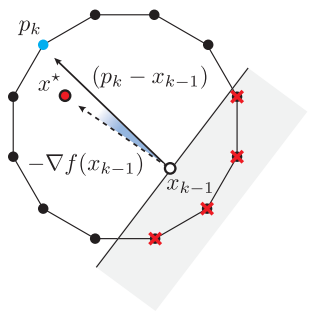


Fig. 1. Schematic illustration of the steepest descent permutation discussed in Observation 2. The black dots with a red cross are the extreme points that are non-descent directions. Frank-Wolfe with $f(x) = (1/2)\|x^\star - x\|_2^2$ chooses the extreme point p_k (i.e., the permutation) that minimizes the angle between $(p_k - x_{k-1})$ and $-\nabla f(x_k) = (x^\star - x_{k-1})$.

and choses a step size $\theta > 0$ such that $f(x_{k-1} + \theta(p_k - x_{k-1})) < f(x_{k-1})$. The essence of the algorithm is that when f is smooth on $\mathcal{B}$,[12] there always exists an extreme point that is a direction in which it is possible to improve the objective function. The step size can be selected in a variety of ways (e.g. constant, line search, etc.) and differently from the previous section, Frank-Wolfe does *not* require that $x_{k-1} + \theta_k p_k \preceq x^\star$ where $x^\star$ is the doubly stochastic matrix we want to decompose. Also, Frank-Wolfe ensures, by construction, that x_k is a convex combination of the permutation matrices throughout the iterations. As objective function, we use $f(x) = (1/2)\|x - x^\star\|_2^2$ to streamline exposition but also because it allows us to make the following observations:

Observation 1 (Weighted Search Direction): For this particular choice of objective function, we have that $\nabla f(x_{k-1}) = -(x^\star - x_{k-1})$. Hence, the update in Eq. (14) becomes

$$p_k \in \arg\min_{u \in \mathcal{P}} -(x^\star - x_{k-1})^T u,$$

which is equivalent to solving the linear program `LP` $(-(x^\star - x_{k-1}), \mathcal{B})$. *That is, computing an extreme point with Frank-Wolfe and* `Birkhoff` *(Algorithm 4) is the same except for the ceiling.*[13] *Note that by ceiling the vector $-(x^\star - x_{k-1})$, we are "weighting" all the components that are not equal to zero equally. Without the ceiling, the Frank-Wolfe update takes into account the geometry of the decomposition, i.e., how close x_{k-1} is to $x^\star$ entry-wise.*

Observation 2 (Steepest Descent Permutation): The extreme points selected by Frank-Wolfe corresponds to obtaining the "steepest" descent direction, or direction $(p_k - x_{k-1})$ that has the smallest angle with respect to $(x^\star - x_{k-1})$. Note that $(x^\star - x_{k-1}) = -\nabla f(x_{k-1})$ is the direction that goes straight to the target value $x^\star$, and that

$$\begin{aligned} &\arg\min_{u \in \mathcal{P}} \nabla f(x_{k-1})^T u \\ &\overset{(a)}{=} \arg\min_{u \in \mathcal{P}} \|\nabla f(x_{k-1})\|_2 \|u\|_2 \cos\phi_{\langle \nabla f, u \rangle} \\ &\overset{(b)}{=} \arg\min_{u \in \mathcal{P}} \ \cos\phi_{\langle \nabla f, u \rangle} \end{aligned}$$

[12]There exists a constant L such that $f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|_2^2$ for all $x, y \in \mathcal{B}$.

[13]Recall also that with FW there is not requirement that $x^\star \succeq x$.

Algorithm 5 Frank-Wolfe (FW) With Quadratic Objective and Line Search

1: As Algorithm 4, but set $x_0 \in \mathcal{P}$ and replace lines ○,☆, ◇ with
2: $p_k \leftarrow \texttt{LP}\ (-(x^\star - x_{k-1}), \mathcal{B})$
3: $\theta_k \leftarrow (x^\star - x_{k-1})^T(p_k - x_{k-1})/\|p_k - x_{k-1}\|_2^2$
4: $x_k \leftarrow x_{k-1} + \theta_k(p_k - x_{k-1})$

Algorithm 6 Fully Corrective Frank-Wolfe (FCFW)

1: As Algorithm 4, but set $x_0 \in \mathcal{P}$ and define $V_0 = \emptyset$. Let Δ_k be the k-simplex. Replace lines ○,☆, ◇ with
2: $p_k \leftarrow \texttt{LP}\ (\nabla f(x_{k-1}), \mathcal{B})$
3: $V_k \leftarrow [V_{k-1}, p_k]$
4: $(\theta_1, \ldots, \theta_k) \leftarrow \arg\min_{u \in \Delta_k}\ \|V_k u - x^\star\|_2^2$
5: $x_k \leftarrow V_k(\theta_1, \ldots, \theta_k)$

where (a) follows from the dot product and (b) since $\|p\|_2 = \sqrt{n}$ for all $p \in \mathcal{P}$, and $\|\nabla f(x_{k-1})\|_2$ does not depend on p. The RHS of the last equation corresponds to maximizing $\cos\phi_{\langle -\nabla f, p\rangle}$, which is equivalent to finding the $p \in \mathcal{P}$ that minimizes the angle between $-\nabla f(x) = (x^\star - x)$ and $(p - x)$. Furthermore, since the Birkhoff polytope is regular and the number of extreme points increases factorially with n, we can expect $\phi_{\langle -\nabla f, p\rangle}$ to be small. Figure 1 shows, schematically, how Frank-Wolfe selects the extreme point that has the smallest angle with respect to $(x^\star - x)$. The black dots with a red cross are "non-descent" permutations that will not improve the decomposition approximation.

Both observations rely on the objective function being quadratic; however, we can expect similar properties for other smooth convex objectives. For example, we could use $f(x) = (x^\star - x)^T Q(x^\star - x)$ where Q is a positive definite matrix that emphasizes which of the components in vector $x^\star - x$ to minimize. In Section VI, we will include a log-barrier function to the objective. In the rest of the section, we will use a quadratic objective function to streamline exposition.

C. Frank-Wolfe With Line Search

The procedure of the Frank-Wolfe algorithm is given in Algorithm 5. Differently from Birkhoff's approach, FW uses an extreme point as a starting point instead of the origin. Note that $0 \notin \mathcal{B}$. The choice of step size is indicated in step 3, and corresponds to carrying out line search. This can be easily verified. Let $x_k := x_{k-1} + \theta_k(p_k - x_{k-1})$ be the k'th iterate, and observe that we can write

$$\begin{aligned}\frac{1}{2}\|x_k - x^\star\|_2^2 - \frac{1}{2}\|x_{k-1} - x^\star\|_2^2 \\ = \frac{1}{2}\|x_{k-1} + \theta_k(p_k - x_{k-1}) - x^\star\|_2^2 - \frac{1}{2}\|x_{k-1} - x^\star\|_2^2 \\ = \theta_k(x_{k-1} - x^\star)^T(p_k - x_{k-1}) + \frac{\theta_k^2}{2}\|p_k - x_{k-1}\|_2^2.\end{aligned}$$

The RHS of the last equation is a quadratic function in θ_k, and its minimizer can be obtained in closed form. And since equality holds in the last equation, minimizing the quadratic function on the RHS is equivalent to minimizing the LHS with line search. Hence, Algorithm 5 corresponds to Frank-Wolfe with line search, and so from [28, Theorem 1][14] we have the bound

$$\|x_k - x^\star\|_2^2 \leq \frac{4n}{k+2}. \tag{15}$$

[14]The bound in Eq. (15) follows from Theorem 1 in [28] with $\delta = 0$ (i.e., in our problem the gradients are noiseless) and $C_f = \max_{u,v\in\mathcal{B}}\|u - v\|_2^2 = 2n$.

Rearranging terms in Eq. (15), we can obtain an upper bound on the sparsity of FW.

Theorem 2 (FW Sparisty): Algorithm 5 obtains an ϵ-approximate decomposition with at most $k \leq 4n/\epsilon^2$ permutation matrices, where $\epsilon = \|x_k - x^\star\|_2$.

The bound in Theorem 2 says that the sparsity of FW is of the order of $O(1/\epsilon^2)$, so we may not be able to obtain sparse decompositions if ϵ is small. One of the issues with first-order-methods such as FW is the zig-zagging phenomenon[15] when the approximate decomposition is close to $x^\star$. Hence, even though FW selects the steepest descent direction, the step size choice (θ_k) is not enough. One way to avoid zig-zagging is to recompute the weights of all the atoms or extreme points discovered so far, which is in essence what the fully corrective variant of the algorithm does.

D. Fully Corrective Frank-Wolfe (FCFW)

This variant of Frank-Wolfe algorithm *recomputes* the weights assigned to vectors $p_1, \ldots, p_k$ in every iteration k. The FCFW procedure is described in Algorithm 6. Like the FW algorithm, FCFW selects a $x_0 \in \mathcal{P}$ and computes a new permutation/extreme point by solving a linear program $\texttt{LP}\ (\nabla f(x_{k-1}), \mathcal{B})$. The permutations are collected in matrix V_k, and the weights $(\theta_1, \ldots, \theta_k)$ are selected to minimize $\|V_k(\theta_1, \ldots, \theta_k) - x^\star\|_2^2$ subject to $\sum_{i=1}^k \theta_i = 1$ and $\theta_i \geq 0$ for all $i = 1, \ldots, k$. Importantly, the (re)computation of weights in Algorithm 6 involves solving a quadratic program (QP) of dimension $i = 1, \ldots, k$. By Theorem 1 in [19], we have the bound

$$\|x_k - x^\star\|_2^2 \leq \|x_0 - x^\star\|_2^2 \exp\left(-\frac{\mu}{4L}\left(\frac{\lambda}{M}\right)^2 k\right), \tag{16}$$

where μ/L is the condition number and $(\lambda/M)^2$ the eccentricity[16] of the Birkhoff polytope. These two parameters are usually not known, however, not in our problem since the Birkhoff polytope and the objective function $f(x) = (1/2)\|x - x^\star\|_2^2$ have remarkable structure. We establish the eccentricity of the Birkhoff polytope in the next lemma.

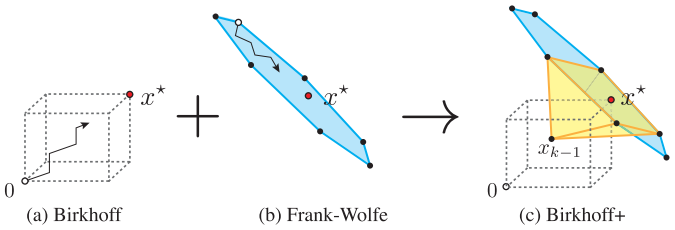
Lemma 4: The eccentricity $(\lambda/M)^2$ of the Birkhoff polytope is lower bounded by $1/(2n^3)$.

Proof: See the Appendix. ∎

Using the last lemma and the fact that the condition number of the objective function (μ/L) is equal to 1, we can obtain the FCFW's sparsity.

[15]See the discussion on page 2 in [19].

[16]The eccentricity of a set is similar to the condition number of a function; see [30, pp. 461].



(a) Birkhoff (b) Frank-Wolfe (c) Birkhoff+

Fig. 2. Schematic illustration of the (a) Birkhoff's and (b) Frank-Wolfe approaches and how they combine into the (c) new setup. The yellow polygon represents the convex hull of x_{k-1} and the permutation matrices in $\mathcal{I}_k(\alpha)$.

Theorem 3 (FCFW Sparsity): Algorithm 6 obtains an ϵ-approximate decomposition with at most $k \leq 8n^3 \log(2n/\epsilon^2)$ permutation matrices, where $\epsilon = \|x_k - x^\star\|_2$.

Proof: See the Appendix. ∎

From the last theorem, we have that the number of extreme points required to obtain an approximate Birkhoff decomposition increases logarithmically with the decomposition error. This is a huge improvement with respect to the sparsity result obtained with the line search FW in Theorem 2. Unfortunately, FCFW is less exciting in practice because recomputing the weights is expensive computationally since the size of the quadratic program (step 4 in Algorithm 6) increases with the number of permutations. Furthermore, the accuracies of the quadratic solvers such as SCS [34], Ipopt [35], and Gurobi [36] are in the order of 10^{-6}, which means that we cannot obtain decompositions with accuracies below 10^{-3}. The latter can be observed in Figure 3a in the numerical evaluation.

VI. New Algorithm

Birkhoff and FCFW algorithms have both logarithmic sparsity, but they are very different algorithmically. On the one hand, weights are easy to compute in Birkhoff's approach,[17] but finding a good permutation matrix is slow as it requires to solve *multiple* linear programs (e.g., [8]). In contrast, FCFW can obtain a good permutation matrix by solving a *single* linear program (see Observation 2), but it requires to solve a quadratic program to (re)calculate the weights.

In this section, we present Birkhoff+ (Algorithm 7), a variation of the original Birkhoff's algorithm that uses the intuition behind Frank-Wolfe to obtain sparse decompositions in a fast manner. The performance of Birkhoff+ is evaluated in Section VII.

A. Approach

The intuition behind our approach is shown schematically in Figure 2. In brief, Birkhoff's algorithm (Figure 2a) can be seen as constructing a path from the origin ($x_0 = 0$) to the target value ($x^\star$) while always remaining in the dotted box (i.e., $x_k \preceq x^\star$ for all $k = 0, 1, 2, \ldots$). Frank-Wolfe (Figure 2b), on the other hand, constructs a path from a permutation matrix $x_0 \in \mathcal{P}$ to the target value $x^\star$ within the polytope of doubly stochastic matrices (blue surface). Our approach (Figure 2c) can be regarded as using Frank-Wolfe within the polytope $\text{conv}(\mathcal{I}_k(\alpha) \cup x_{k-1})$ (yellow polygon in Figure 2c) with the additional constraint that the approximate decomposition must be within the dotted box. That is, we want to use the path or permutations that Frank-Wolfe would select while remaining in the box that characterizes the Birkhoff's approach. It is important to use $\text{conv}(\mathcal{I}_k(\alpha) \cup x_{k-1})$ instead of $\text{conv}(\mathcal{P} \cup x_{k-1})$ (i.e., all permutations) as the algorithm may otherwise not converge. The latter is shown formally in the following theorem.

Theorem 4: Consider Algorithm 4 and replace line ∘ with LP$(-(x^\star - x_{k-1}))$. Then, there may not exist a k for which $\|x_k - x^\star\|_2^2 \leq \epsilon$ for any $\epsilon > 0$.

We can prove the theorem by example. Suppose we want to decompose the following $n \times n$ doubly stochastic matrix

$$\frac{1}{n-1}\begin{bmatrix} n-2 & 0 & \cdots & 0 & 1 \\ 0 & n-2 & & & 1 \\ \vdots & & \ddots & & \vdots \\ 0 & & & n-2 & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}. \tag{17}$$

That is, (i) the first $n-1$ entries in the diagonal are equal to $(n-2)/(n-1)$, (ii) the first $n-1$ entries of the last row are equal to $1/(n-1)$, and (iii) the first $n-1$ entries of the last column are equal to $1/(n-1)$. Note the sum of each row and column is equal to one. Next, suppose that $f(x) = (1/2)\|x^\star - x\|_2^2$ where $x^\star$ is the matrix in Eq. (17) in column form. In the first iteration ($x_0 = 0$), Frank-Wolfe selects a permutation by solving the linear program LP$(-x^\star, \mathcal{B})$, the solution of which is the identity matrix since the doubly stochastic matrix in Eq. (17) is diagonally dominant. And because the last entry of the matrix in Eq. (17) is equal to zero, we have that BIRKHOFF_STEP$(x^\star, x_{k-1}, p_k) = 0$ and therefore $x_k = x_{k-1}$. That is, the algorithm will be "stuck."

In sum, a Birkhoff-type algorithm that selects permutation matrices with Frank-Wolfe using all the permutation matrices $\mathcal{P}$ may not converge. However, we can use Frank-Wolfe with the permutations in the set $\mathcal{I}_k(\alpha)$, which ensures not only that the algorithm converges but that this has logarithmic sparsity (Theorem 1).

1) Objective Function With Barrier: Since Birkhoff's approach restricts x_k to remain in the Birkhoff's dotted box (see Figure 2), it is reasonable to use an objective function that aims to construct a path to $x^\star$ from within the box. For that, we define

$$f_\beta(x) \;= f(x) - \beta \sum_{j=1}^{d} \log(x^\star(j) - x(j) + \epsilon/d), \tag{18}$$

where $\beta \geq 0$ and $x(j)$ is the j'th component of vector x. Note that f_β is convex as this is the composition of f plus a convex penalty/barrier function $-\beta \sum_{j=1} \log(x^\star(j) - x_{k-1}(j) + \epsilon/d)$. The term ϵ/d in the barrier is used for numerical stability as otherwise the barrier goes to $+\infty$ when $x^\star(j) = x(j)$. The motivation for using a barrier function comes from interior point methods in optimization, where parameter β is typically tuned throughout the algorithm to allow $x_k \to x^\star$. Note that $f_\beta \to f$ as $\beta \to 0$.

[17] Birkhoff's step size requires to find the smallest of n elements.

Algorithm 7 `Birkhoff+`

1: As Algorithm 4, but take $\beta \geq 0$ also as input. Replace line $\circ$ with
2: $\alpha \leftarrow (1 - \sum_{i=1}^{k-1} \theta_i)/n^2$
3: $p_k \leftarrow \texttt{LP}(\nabla f_\beta(x_{k-1}), \text{conv}(\mathcal{I}_k(\alpha)))$

Algorithm 8 `Birkhoff+(max_rep)` — With Permutation Selection Refinement

1: As Algorithm 7, but replace line $\circ$ with
2: **for** $i = 1, \ldots,$ `max_rep` **do**
3: $p_i \leftarrow \texttt{LP}(\nabla f_\beta(x_{k-1}), \text{conv}(\mathcal{I}_k(\alpha)))$
4: $\theta_i \leftarrow \texttt{BIRKHOFF\_STEP}(x^\star, x_{k-1}, p_k)$
5: **if** $(\theta_i > \alpha)$ $\alpha \leftarrow \texttt{BIRKHOFF\_STEP}(x^\star, x_{k-1}, p_k)$
6: **else** exit while loop
7: $p_k \leftarrow p_i$
8: **end for**

B. `Birkhoff+` Algorithm Description and Complexity

The procedure of `Birkhoff+` is described in Algorithm 7, and consists of replacing how permutation matrices are selected in Algorithm 4 with $\texttt{LP}(\nabla f_\beta(x_{k-1}), \text{conv}(\mathcal{I}_k(\alpha)))$, where f_β is as defined in Eq. (18). Parameter β can be selected to emphasize the barrier over the objective function f. In our case, we do not need $\beta \to 0$ as by selecting permutations from $\mathcal{I}_k(\alpha)$ is enough to allow the algorithm to make progress. The convergence of the algorithm is stated formally in the following corollary.

Corollary 2: Algorithm 7 obtains an ϵ-approximate decomposition with at most $k \leq O(\log(1/\epsilon))$ permutation matrices.

The complexity of `Birkhoff+` per iteration is equal to solving a linear program with a simplex type method. The linear program $\texttt{LP}(\nabla f_\beta(x_{k-1}), \text{conv}(\mathcal{I}_k(\alpha)))$ can be carried out with $\texttt{LP}(\nabla f_\beta(x_{k-1}) + b_k, \mathcal{B})$ where $b_k = d/\epsilon \cdot \mathbb{I}_{\{0,1\}}(x^\star - x_{k-1} \preceq \alpha)$ is a penalty vector to force the solver to do not select the components of vector $(x^\star - x_k)$ smaller than α.

Finally, we note that `Birkhoff+` depends on how we define set $\mathcal{I}_k(\alpha)$. Algorithm 8 is a meta-heuristic for selecting α based on Birkhoff's step size. In particular, α is set to $(1 - \sum_{i=1}^{k} \theta_i)/n^2$ in the first iteration and then equal to the largest step size for the permutation selected using the Frank-Wolfe-type update. The search for a large α terminates when the maximum number of repetitions (`max_rep`) is reached or the value of α does not increase. We call Algorithm 8 `Birkhoff+(#)`, where `#` indicates the maximum number of permutation refinements. `Birkhoff+(1)` is equivalent to `Birkhoff+` as it computes only one permutation matrix.

VII. Numerical Evaluation

In this section, we evaluate performance of `Birkhoff+` and compare it to existing algorithms. Our goal is to illustrate the algorithms' characteristics and how different traffic matrices affect the performance of a circuit switch in terms of throughput, configurations computation time, and number of configurations. The code of `Birkhoff+` is available as a Julia package in [37].

A. Setup

The `Birkhoff`, `FW`, `FCFW`, `Birkhoff+` and `Birkhoff+(#)` algorithms are implemented in Julia [38] and as indicated in Algorithms 4, 5, 6, 7 and 8 respectively. Parameter β is fixed to 1 and the maximum number of permutation refinements in `Birkhoff+(#)` to 10 — however, we observe in the experiments that the actual number of permutation refinements is usually less than 3. `Solstice` corresponds to Algorithm 2 in [8], and `Eclipse` to Algorithm 2 in [9]. The linear programs $\texttt{LP}(\cdot,\cdot)$ are carried out with Clp [39] in all algorithms and return an extreme point/permutation matrix. The quadratic programs in the `FCFW` algorithm are carried out with Ipopt [35]. Both solvers are open-source.

Traffic demand matrices are generated by sampling permutations uniformly at random, and weights are selected to model the type of load in data centers. In particular, we follow the evaluation scenario in [9], where traffic matrices are sparse and consist of 12 flows. Three of the flows are large and carry the 70% of the load, while the rest are small flows and carry the remaining 30% of the traffic. We note that the traffic matrix in practical scenarios may be below the switch's capacity (i.e., the sum of each row or column may be smaller than one), and so we first need to add a virtual load to the traffic matrix to make it doubly stochastic.[18] For simplicity, we assume in the evaluation that the demand matrices are doubly stochastic.

Finally, the numerical evaluation is carried out on a computer equipped with an Intel i7 8700B (3.2 GHz) CPU and 32 GB of memory. The version of Julia is 1.3.1.

B. Experiments

We first study the algorithms' characteristics, and then show how those affect the performance of a circuit switch.

1) Decomposition Approximation vs. Number of Permutations and Time: We set $n = 32$ and sample traffic matrices as indicated in Section VII-A. Also, we fix $\epsilon = 10^{-4}$, $k_{\max} = 300$ and $\delta = 10^{-2}$ (just for `Eclipse`).[19] Figure 3 shows the algorithms decomposition error in terms of permutations and time. The results are the average of 50 realizations.

Observe from Figure 3a that the decomposition error of `Birkhoff` is large until it converges exactly in the last iteration ($k \approx 250$). On the other hand, `FW` progresses quickly, but it slows down drastically around $\epsilon = 0.9$. The latter is due to the $O(1/\epsilon^2)$ sparsity rate and the zig-zagging phenomenon typical in first-order-methods (see Section V-C). The `FCFW` has a better sparsity performance than `FW`, but it cannot obtain decomposition with an ϵ below $0.5 \cdot 10^{-3}$ due to the numerical accuracy of the quadratic solvers (see Section V-D). `Eclipse` has a better performance than previous algorithms until it gets stuck between $\epsilon \in [10^{-2}, 10^{-1}]$.

[18]The work in [8] (see Section 4.2.1) uses the term "stuffing" for adding virtual load to the traffic matrix. Stuffing can be seen as a special type of projection of the demand matrix onto the Birkhoff polytope. Technically, for a demand matrix D, we need to find a matrix $S \in \mathcal{B} - D$. Matrix S may not be unique and finding the best virtual load matrix for our algorithm is an interesting problem but out of the scope of the paper.

[19]The value corresponds to having a switching cost of 10 ms.

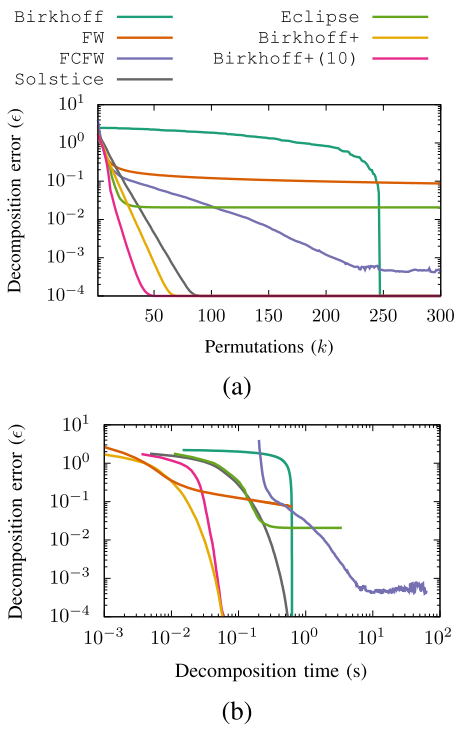


Fig. 3. Decomposition error (ϵ) of `Birkhoff`, `FW`, `FCFW`, `Solstice`, `Eclipse`, `Birkhoff+`, and `Birkhoff+(10)` algorithms depending on (a) the number or permutations and (b) time. The figure shows the average of 50 realizations.

We conjecture the latter is because `Eclipse` selects permutations using a Max-Weight-type matching, and so it may face similar issues as when we combine Frank-Wolfe and Birkhoff approaches directly; see discussion in Section VI-A. Also, the performance guarantees of `Eclipse` given in [9] are for the problem type in [24] (see Section II) and not for decomposing a doubly stochastic matrix. Finally, observe that `Solstice`, `Birkhoff+`, and `Birkhoff+(10)`[20] have all better sparsity performance than the previous algorithms and that `Birkhoff+(10)` is noticeably better for $\epsilon < 0.1$. The last three algorithms have linear convergence/logarithmic sparsity (y-axis is in log-scale) but different condition numbers $(1-\mu_{\min})$: 0.89, 0.85 and 0.82 respectively.[21] Recall the condition number indicates how an additional permutation reduces the decomposition error multiplicatively (see discussion in Section IV-C).

Figure 3b shows the decomposition error against the running time. Observe that `Birkhoff+` is the fastest followed by `Birkhoff+(10)`. `FW` is also fast for $\epsilon > 0.1$, but it slows down afterward for the same reason explained above. `Solstice` and `Eclipse` are both slower than `Birkhoff+` by an order of magnitude since they need to solve multiple linear programs to select a permutation matrix. The running time of `Birkhoff` is in line with the sparsity results: it makes slow progress until it converges exactly in the last iteration. Finally, `FCFW` is the slowest as it has to recompute all the weights (i.e., solve a quadratic program) every time it adds a new permutation to the decomposition.

2) Circuit Switch Performance: We now evaluate the algorithm's performance when used to compute the switching configurations for a circuit switch with $n = 100$ ports. The performance metrics we evaluate are the throughput, the configurations computation time, and the number of switching configurations. We carry out three experiments where we vary the reconfiguration cost, the skewness and sparsity of the traffic matrix, and the configurations computation overhead. Importantly, now the traffic matrix $X^\star$ is associated with a time window W that enforces the decomposition to satisfy $\sum_{i=1}^{k}(\theta_i+\delta) \leq W$, i.e., the time spent transmitting $(\sum_{i=1}^{k}\theta_i)$ and reconfiguring (δk) cannot exceed the time window duration (W). Finally, we only evaluate `Solstice`, `Eclipse`, `Birkhoff+`, and `Birkhoff+(10)` as (i) `Birkhoff` and `FW` have a poor performance, and (ii) `FCFW` is very slow when $n \geq 32$ (see times in Figure 3b).

Experiment 1 (Impact of Reconfiguration Time): Figure 4 shows the algorithms' performance in terms of throughput, running time, and the number of configurations depending on the ratio δ/W (the impact of the reconfiguration delay proportionally to the time window duration). Observe from the figure that `Birkhoff+(10)` outperforms the other algorithms in terms of throughput. For instance, for $\delta/W = 10^{-2}$, `Birkhoff+(10)` achieves a 7% more throughput than `Eclipse` and `Solstice`. `Birkhoff+` has almost the same throughput than `Eclipse` and `Solstice`. Regarding the time required to compute the switching configurations, `Solstice` and `Eclipse` are slower than `Birkhoff+` and `Birkhoff+(10)` by an order of magnitude; however, the difference decreases as δ/W increases because we have fewer switching configurations as a result of larger reconfiguration penalties (c.f. Figure 4b and Figure 4c). Finally, observe from Figure 4c that `Birkhoff+(10)` can obtain decompositions with half of the configurations compared to other algorithms when ϵ is small (i.e., 10^{-4}). **Conclusions:** `Birkhoff+` has the same performance in terms of throughput and number of switching configurations than `Solstice` and `Eclipse`, but it is 10 times faster. `Birkhoff+(10)` obtains higher throughput than all algorithms and it is only slightly slower than `Birkhoff+`.

Experiment 2 (Sparsity and Skewness): Now we set $\delta/W = 10^{-2}$ and evaluate the algorithms' performance depending on the skewness and the sparsity of the demand matrix. In Figure 5, we vary the fraction of the load carried by the small flows. Observe that as before, `Birkhoff+(10)` outperforms the other algorithms, and that `Birkhoff+`, `Solstice`, and `Eclipse` are almost the same in terms of throughput for different demand matrices. Furthermore, there is little variation on the running time and the number of switching configurations—despite a slight bend in the curves when the traffic matrix contains the same fraction of large and small flows.

In Figure 6, we show the results when we vary the number of permutations used to generate the demand matrix. Each permutation matrix is sampled as explained in Section VII-A. Observe from the figure that the sparsity of the traffic matrix

[20]The number in the parentheses is the maximum number of permutations refinements (`max_rep`).

[21]Average of the 50 first iterations.

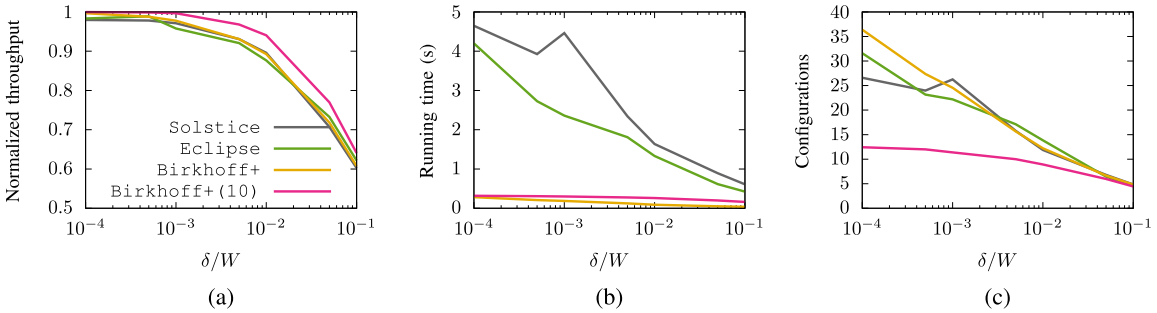


Fig. 4. Circuit switch performance (throughput, running time, and number of configurations) depending on δ/W, where δ is the switching time and W the time window duration. The figures show the average of 50 realizations.

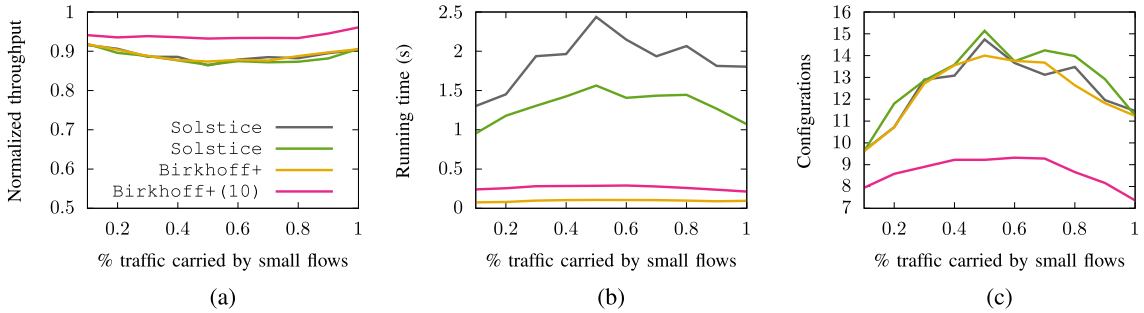


Fig. 5. Circuit switch performance (throughput, running time, and number of configurations) depending on the load carried by the small flows. The figures show the average of 50 realizations.

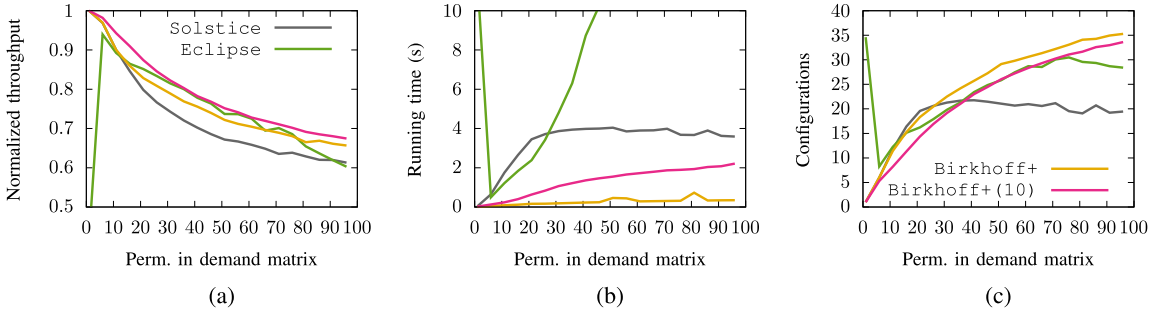


Fig. 6. Circuit switch performance (throughput, running time, and number of configurations) depending on the number of permutations matrices used to generate the traffic matrix. The figures shows the average of 50 realizations.

has a significant impact on the throughput, running time, and the number of the switching configurations. In particular, observe from Figure 6a that the throughput of all algorithms decreases and that `Eclipse` is comparable to `Birkhoff+(10)` as the traffic matrix becomes denser. However, the running time of `Birkhoff+(10)` does not explode (see Figure 6b) and `Birkhoff+(10)` does not get stuck when the traffic demand matrix is very sparse.[22] Regarding `Birkhoff+`, observe that now the running times difference with `Birkhoff+(10)` becomes more noticeable as the demand matrix becomes denser. Finally, observe from Figure 6c that the number of switching configurations increases with the density of the traffic matrix for all algorithms. **Conclusions:** The skewness of the demand matrix has little impact on to the performance of all algorithms. The sparsity, on the other hand, plays an important role. `Eclipse` has a similar performance than `Birkhoff+(10)`, but it is notably slower.

Experiment 3 (Configurations Computation Overhead): This experiment shows how time to compute the switching configurations affects the circuit switch's throughput. In particular, we set $\delta/W = 10^{-2}$ and truncate the decomposition to satisfy $\sum_{i=1}^{k}(\theta_i + \delta) \leq W - T$, where T is the time to compute the switching configurations. The values of T for this particular setting are given in Figure 4b. Figure 7 shows the throughput for different values of W in seconds. Observe from the figure that the throughput increases with W for all algorithms. When W is small (i.e., the decomposition computation overhead is large), `Birkhoff+` has a higher throughput than `Birkhoff+(10)` because it is faster—recall `Birkhoff+`

[22]See discussion in Section VII-B.1.

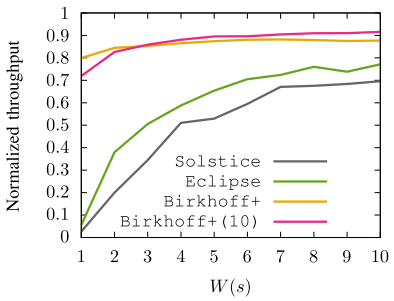


Fig. 7. Circuit switch throughput when the time to compute the switching configurations is an overhead. The figure shows the average of 50 realizations.

selects a new switching configuration by solving a single linear program. However, `Birkhoff+(10)`'s throughput is higher when $W > 5$ since the reconfiguration time is larger than the time to compute the switching configurations. Regarding `Eclipse` and `Solstice`, observe that both are affected heavily by the decomposition overhead. For instance, when $W = 5$, `Birkhoff+` has 67% and 34% more throughput than `Solstice` and `Eclipse` respectively. Also, note that when $W = 1$, `Birkhoff+` can serve 80% of the traffic whereas `Solstice` and `Eclipse` almost nothing. **Conclusions:** `Birkhoff+` outperforms `Solstice` and `Eclipse` and it is slightly better than `Birkhoff+(10)` when the time windows are short. As with the reconfiguration costs, the benefit of computing switching configurations fast diminishes as the time window duration increases.

VIII. Conclusion

This paper studies how to compute switching configurations for circuit switches. We have revisited Birkhoff's approach and established its properties in terms of the number of switching configurations required to obtain an approximate representation of a traffic matrix. A new algorithm (`Birkhoff+`) is proposed, which obtains representations with fewer switching configurations than previous work (`Solstice`, `Eclipse`) and is 10-100 times faster depending on the setting. The latter is important in terms of throughput when traffic bursts are short-lived, and so the time required to compute the switching configurations is a non-negligible overhead. We also propose a variant of `Birkhoff+` that is slightly slower but obtains representations with even fewer switching configurations. The performance of the proposed algorithms is evaluated through exhaustive numerical experiments for traffic demand matrices that capture traffic characteristics in data centers.

Appendix

A. Proofs of Section IV

We start by presenting two lemmas. The first lemma gives an upper bound on the Frobenius norm of a doubly stochastic matrix.

Lemma 5: $\|X\|_F \leq \sqrt{n}$ *for any doubly stochastic matrix* X.

Proof: Let r_i be the i'th row of X and note $\|r_i\|_1 = 1$ for all $i \in \{1, \ldots, n\}$, i.e., the sum of a row is equal to 1. Observe

$$\|X\|_F = \sqrt{\mathrm{Tr}\,(XX^*)} = \sqrt{\sum_{i=1}^{n} \|r_i\|_2^2} \leq \sqrt{\sum_{i=1}^{n} \|r_i\|_1^2} \leq \sqrt{n},$$

where the first inequality follows because $\|\cdot\|_2 \leq \|\cdot\|_1$. ■

The second lemma establishes that $X^\star - X_k$ is a scaled doubly stochastic matrix.

Lemma 6: Let $X_k = \sum_{i=1}^{k} \theta_i P_i$ *and suppose* $X_k(a,b) \leq X^\star(a,b)$ *for all* $a, b \in \{1, \ldots, n\}$ *and* $k \geq 1$. *Then,*

(a) $\dfrac{X^\star - X_k}{1 - \sum_{i=1}^{k} \theta_i}$ *is doubly stochastic*

(b) $\|X^\star - X_k\|_F \leq \sqrt{n}\left(1 - \sum_{i=1}^{k} \theta_i\right)$

Proof: We start with (a). By assumption, $0 \leq X_k(a,b) \leq X^\star(a,b) \leq 1$ for all $a, b \in \{1, \ldots, n\}$. Hence, we only need to show that the sum of each row and column is equal to one. Observe

$$\begin{aligned}
&(1 - \textstyle\sum_{i=1}^{k} \theta_i)^{-1}(X^\star - X_k)\mathbf{1} \\
&\quad = (1 - \textstyle\sum_{i=1}^{k} \theta_i)^{-1}(X^\star - \textstyle\sum_{i=1}^{k} \theta_i P_i)\mathbf{1} \\
&\quad = (1 - \textstyle\sum_{i=1}^{k} \theta_i)^{-1}(X^\star\mathbf{1} - \textstyle\sum_{i=1}^{k} \theta_i P_i\mathbf{1}) \\
&\quad = (1 - \textstyle\sum_{i=1}^{k} \theta_i)^{-1}(\mathbf{1} - \mathbf{1}\textstyle\sum_{i=1}^{k} \theta_i) \\
&\quad = \mathbf{1}(1 - \textstyle\sum_{i=1}^{k} \theta_i)^{-1}(1 - \textstyle\sum_{i=1}^{k} \theta_i) \\
&\quad = \mathbf{1}
\end{aligned}$$

The same argument above can be used to show that $(1 - \sum_{i=1}^{k} \theta_i)^{-1}\mathbf{1}^T(X^\star - X_k) = \mathbf{1}^T$, i.e., the sum of each column is equal to one.

For (b), observe $\|(1 - \sum_{i=1}^{k} \theta_i)^{-1}(X^\star - X_k)\|_F = (1 - \sum_{i=1}^{k} \theta_i)^{-1}\|X^\star - X_k\|_F \leq \sqrt{n}$ by Lemma 5. Rearranging terms yields the result. ■

Proof of Lemma 2

We start by proving the lower bound. We first note $\|X\|_F \geq 1$ for any doubly stochastic matrix X. Recall

$$\|X\|_F \geq \|X\|_2 := \sup\left\{\frac{\|Xu\|_2}{\|u\|_2} \text{ with } u \in \mathbf{R}^n \text{ s.t. } u \neq 0\right\}$$

Let $u = \mathbf{1}$ in the equation above to obtain

$$\|X\|_F \geq \frac{\|X\mathbf{1}\|_2}{\|\mathbf{1}\|_2} = \frac{\|\mathbf{1}\|_2}{\|\mathbf{1}\|_2} = 1,$$

where $X\mathbf{1} = \mathbf{1}$ follows since X is doubly stochastic. Next, since $\frac{X^\star - X_k}{1 - \sum_{i=1}^{k} \theta_i}$ is doubly stochastic by Lemma 6, we have

$$1 \leq \left\|\frac{X^\star - X_k}{1 - \sum_{i=1}^{k} \theta_i}\right\|_F = \left(1 - \sum_{i=1}^{k} \theta_i\right)^{-1} \|X^\star - X_k\|_F$$

Rearranging terms yields the lower bound.

For the upper bound, observe

$$\begin{aligned}
&\|X_k - X^\star\|_F^2 \\
&\text{(a)} = \|X_{k-1} + \theta_k P_k - X^\star\|_F^2 \\
&\quad = \|X_{k-1} - X^\star\|_F^2 + \theta_k^2\|P_k\|_F^2 \\
&\qquad + 2\theta_k \textstyle\sum_{a,b} P_k(a,b)(X_{k-1}(a,b) - X^\star(a,b))
\end{aligned}$$

$$\begin{aligned}
\text{(b)} &\leq \|X_{k-1} - X^\star\|_F^2 + \theta_k^2 \|P_k\|_F^2 - 2\theta_k^2 \textstyle\sum_{a,b} P_k(a,b)^2 \\
&= \|X_{k-1} - X^\star\|_F^2 + \theta_k^2 \|P_k\|_F^2 - 2\theta_k^2 n \\
\text{(c)} &\leq \|X_{k-1} - X^\star\|_F^2 + \theta_k^2 n - 2\theta_k^2 n \\
&= \|X_{k-1} - X^\star\|_F^2 - \theta_k^2 n \qquad (19)
\end{aligned}$$

where (a) follows by Algorithm 1, (b) by Eq. (3), and (c) by Lemma 5. Hence,

$$\|X_k - X^\star\|_F^2 \leq \left(1 - \frac{n\theta_k^2}{\|X_{k-1} - X^\star\|_F^2}\right) \|X_{k-1} - X^\star\|_F^2$$

Applying the argument recursively from $i = 1, \ldots, k$

$$\|X_k - X^\star\|_F^2 \leq \|X_0 - X^\star\|_F^2 \prod_{i=1}^{k} \left(1 - \frac{n\theta_i^2}{\|X_{i-1} - X^\star\|_F^2}\right)$$

Finally, since $X_0 = \{0\}^{n\times n}$ and $\|X^\star\|_F \leq \sqrt{n}$ by Lemma 5,

$$\|X_k - X^\star\|_F^2 \leq n \prod_{i=1}^{k} \left(1 - \frac{n\theta_i^2}{\|X_{i-1} - X^\star\|_F^2}\right)$$

Taking square roots on both sides yields Eq. (7).

To conclude, we show that $\theta_i \leq \frac{1}{\sqrt{n}} \|X_{i-1} - X^\star\|_F$ for all $i = 1, 2, \ldots, k$. From Eq. (19), $0 \leq \|X_{k-1} - X^\star\|_F^2 - \theta_k^2 n$. Rearranging terms and taking square roots on both sides completes the proof.

Proof of Lemma 3

Since $n\theta_i^2 / \|X_{i-1} - X^\star\|_F^2 \geq \mu_{\min}$ by assumption, the upper bound in Lemma 2 becomes $\|X_k - X^\star\|_F \leq \sqrt{n}\,(1 - \mu_{\min})^{k/2}$. Next, let $\epsilon = \|X_k - X^\star\|_F$ and write $\epsilon \leq \sqrt{n}\,(1 - \mu_{\min})^{k/2}$. Rearranging terms yields

$$\left(\frac{1}{1 - \mu_{\min}}\right)^{k/2} \leq \frac{\sqrt{n}}{\epsilon}.$$

Taking logs on both sides and further rearranging terms yields the result.

Proof of Theorem 1

We start by showing that we can design a subroutine `PERM` that returns a permutation with an associated weight that is uniformly lower bounded and satisfies the conditions in Eqs. (3)–(5). We have the following lemma.

Lemma 7: Set $\mathcal{I}_k(\alpha)$ with $\alpha = \frac{1 - \sum_{i=1}^{k-1} \theta_i}{(n-1)^2 + 1}$ is non-empty.

Proof: By Lemma 6, $\frac{X^\star - X_{k-1}}{1 - \sum_{i=1}^{k-1} \theta_i}$ is doubly stochastic, and so, by Carathéodory's theorem, we can write it as the convex combination of $(n-1)^2 + 1$ permutation matrices, i.e.,

$$\frac{X^\star - X_{k-1}}{1 - \sum_{i=1}^{k-1} \theta_i} = \sum_{j=1}^{(n-1)^2+1} \beta_j P_j$$

where $\beta_j \geq 0$ and $\sum_{j=1}^{(n-1)^2+1} \beta_j = 1$. Next, note that since the permutation matrices and weights are non-negative, we have that

$$\beta_j P_j(a,b) \leq \frac{X^\star(a,b) - X_{k-1}(a,b)}{1 - \sum_{i=1}^{k-1} \theta_i} \qquad (20)$$

holds for all $a, b \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, (n-1)^2 + 1\}$. Furthermore, since $\sum_{j=1}^{(n-1)^2+1} \beta_j = 1$, we have that

$$\beta_j \geq \frac{1}{(n-1)^2 + 1} \qquad (21)$$

for at least one $j \in \{1, \ldots, (n-1)^2 + 1\}$. Let $\alpha = \beta_j$ such that the last equation holds. Combining Eq. (20) and Eq. (21), we obtain that

$$\frac{1 - \sum_{i=1}^{k-1} \theta_i}{(n-1)^2 + 1} P(a,b) = \alpha P(a,b) \leq X^\star(a,b) - X_{k-1}(a,b)$$

That is, there exists at least a permutation P such that $X_{k-1} + \alpha P(a,b) \leq X^\star(a,b)$, and so set $\mathcal{I}_k(\alpha)$ is non-empty. ∎

We are now in position to present the proof of Theorem 1. By Lemma 7, set $\mathcal{I}_k(\alpha')$ with $\alpha' = \frac{1 - \sum_{i=1}^{k-1} \theta_i}{(n-1)^2+1}$ is non-empty. Now, observe that since $\alpha = \sqrt{\frac{\mu_{\min}}{n}}(1 - \sum_{i=1}^{k} \theta_i) \leq \alpha'$ because $\mu_{\min} = 1/n^3$, we have that $\mathcal{I}_k(\alpha') \subseteq \mathcal{I}_k(\alpha)$ and so $\mathcal{I}_k(\alpha)$ is non-empty. The rest of the proof follows as in Lemma 3 with $\mu_{\min} = \min_{i \in \{1,\ldots,k\}} n\theta_i^2 / \|X_{i-1} - X^\star\|_F^2$.

B. Proofs of Section V

Birkhoff Polytope Representation in Vector Form: The Birkhoff polytope is the set that contains all doubly stochastic matrices. Recall we say that a nonnegative matrix is doubly stochastic if the sum of its rows and columns is equal to one. This corresponds to having $2n$ equality constraints. We can express these in vector form by defining matrices

$$A'(in + 1, in + j) = \begin{cases} 1, & i = 0, \ldots, n-1,\ j = 1, \ldots, n \\ 0, & \text{otherwise} \end{cases}$$

$$A''(j + in, j) = \begin{cases} 1, & i = 0, \ldots, n-1,\ j = 1, \ldots, n \\ 0, & \text{otherwise} \end{cases}$$

and then collecting them in $A = [A'; A'']$. Next, define $b \in \{1\}^{2n}$. Any vector from $\mathbf{R}_+^d$ such that $Ax = b$ correspond to having doubly stochastic matrix in vector form.

For example, with $n = 3$ we have

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Proof of Lemma 4

The eccentricity consists of two parameters. The diameter of the polytope (M) and its pyramidal width (λ). The diameter of the Birkhoff polytope is the maximum distance between two points in $\mathcal{B}$, which is the maximum distance between two vertices. Specifically, this is equal to $\|p - p'\|_2 = \sqrt{2n}$ where $p, p' \in \mathcal{P}$ are two vertices such that $p^T p' = 0$, i.e. have ones in different components.

It is possible to obtain a lower bound on the pyramidal width of the Birkhoff polytope by using the fact that its extreme

points are a subset of the extreme points of the unit cube in d dimensions. Formally, $\mathcal{P} \subset \{0,1\}^d$ and so $\text{conv}(\mathcal{P}) := \mathcal{B} \subset \mathcal{C} := \text{conv}(\{0,1\}^d)$. The latter means that the unit cube is "extreme-point-wise denser" than the Birkhoff polytope and so it has smaller pyramidal width. From Lemma 4 in [19] we can obtain that the pyramidal width of the Birkhoff polytope is lower bounded by $1/\sqrt{d} = 1/n$.[23] Hence, $(\lambda/M)^2 \geq 1/(2n^3)$ as claimed.

Proof of Theorem 3

This theorem is an application of Theorem 1 in [19] with the quadratic objective function $f(x) = (1/2)\|x - x^\star\|_2^2$ and set $\mathcal{B}$. This theorem says that

$$\|x_k - x^\star\|_2^2 \leq \|x_0 - x^\star\|_2^2 \exp\left(-\frac{\mu}{4L}\left(\frac{\lambda}{M}\right)^2 \; k\right)$$

The term $\|x_0 - x^\star\|_2$ can be upper bounded by $\sqrt{2n}$, which is the maximum Euclidean distance between two points in $\mathcal{B}$ (see the proof of Lemma 4). The condition number μ/L is equal to 1 because the objective function is quadratic and $(\lambda/M)^2 \geq 1/(2n^3)$ by Lemma 4. Hence,

$$\|x_k - x^\star\|_2^2 \leq 2n \exp\left(-\frac{k}{8n^3}\right).$$

To conclude, let $\epsilon^2 = \|x_k - x^\star\|_2^2$ and write $\epsilon^2 \leq 2n\exp(-k/(8n^3))$. By expressing k as a function of ϵ in the last equation, we obtain the stated result.

ACKNOWLEDGMENT

Víctor Valls would like to thank Ehsan Kazemi (Yale Institute for Network Science) for many helpful conversations and the counter-example in the proof of Theorem 4.

REFERENCES

[1] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, 2010, pp. 339–350.

[2] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 498–511, Apr. 2014.

[3] X. S. Huang, X. S. Sun, and T. S. E. Ng, "Sunflow: Efficient optical circuit scheduling for coflows," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2016, pp. 297–311.

[4] G. Wang *et al.*, "C-through: Part-time optics in data centers," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2010, pp. 327–338.

[5] H. Liu *et al.*, "Circuit switching under the radar with reactor," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 1–15.

[6] *Polatis 7000*. Accessed: Mar. 25, 2021. [Online]. Available: https://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuit-switch-sdn-enabled.asp

[7] Calient. *Calient 160*. Accessed: Mar. 25, 2021. [Online]. Available: https://www.calient.net/products/s-series-photonic-switch/

[8] H. Liu *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *Proc. 11th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2015, pp. 1–13.

[9] S. Bojja Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathéodory theorems," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Sci.*, 2016, pp. 75–88.

[10] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.

[23] Recall that $d = n^2$.

[11] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 123–137.

[12] R. Kapoor, A. C. Snoeren, G. M. Voelker, and G. Porter, "Bullet trains: A study of NIC burst behavior at microsecond timescales," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2013, pp. 133–138.

[13] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[14] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von neumann input buffered crossbar switches," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2000, pp. 1614–1623.

[15] F. Dufossé and B. Uçar, "Notes on Birkhoff–von Neumann decomposition of doubly stochastic matrices," *Linear Algebra Appl.*, vol. 497, pp. 108–115, May 2016.

[16] A. Livshits and S. Vargaftik, "LUMOS: A fast and efficient optical circuit switch scheduling technique," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 2028–2031, Oct. 2018.

[17] D. Birkhoff, "Tres observaciones sobre el algebra lineal," *Universidad Nacional de Tucuman Revista, Serie A*, vol. 5, pp. 147–151, 1946.

[18] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logistics Quart.*, vol. 3, nos. 1–2, pp. 95–110, 1956.

[19] S. Lacoste-Julien and M. Jaggi, "On the global linear convergence of Frank-Wolfe optimization variants," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 496–504.

[20] J. Von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contrib. Theory Games*, vol. 2, pp. 5–12, Sep. 1953.

[21] M. Marcus, "Some properties and applications of doubly stochastic matrices," *Amer. Math. Monthly*, vol. 67, no. 3, pp. 215–221, 1960.

[22] R. A. Brualdi, "Notes on the Birkhoff algorithm for doubly stochastic matrices," *Can. Math. Bull.*, vol. 25, no. 2, pp. 191–199, Jun. 1982.

[23] J. Kulkarni, E. Lee, and M. Singh, "Minimum Birkhoff-von Neumann decomposition," in *Integer Programming and Combinatorial Optimization*. Cham, Switzerland: Springer, 2017, pp. 343–354. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-59250-3_28

[24] X. Li and M. Hamdi, "On scheduling optical packet switches with reconfiguration delay," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 7, pp. 1156–1164, Sep. 2003.

[25] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. Press, 1963.

[26] R. Schwartz, M. Singh, and S. Yazdanbod, "Online and offline greedy algorithms for routing with switching costs," 2019, *arXiv:1905.02800*. [Online]. Available: http://arxiv.org/abs/1905.02800

[27] S. Bubeck, "Introduction to online optimization," Princeton Univ., Princeton, NJ, USA, Lect. Notes, 2011, vol. 2. Accessed: Jul. 17, 2021. [Online]. Available: http://sbubeck.com/BubeckLectureNotes.pdf

[28] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *Proc. 30th Int. Conf. Mach. Learn.*, Jun. 2013, pp. 427–435.

[29] F. Dufossé, K. Kaya, I. Panagiotas, and B. Uçar, "Further notes on Birkhoff–von Neumann decomposition of doubly stochastic matrices," *Linear Algebra Appl.*, vol. 554, pp. 68–78, Oct. 2018.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[31] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, Jun. 2015.

[32] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, Jan. 2012.

[33] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Englewood Cliffs, NJ, USA: Prentice-Hall 1989.

[34] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *J. Optim. Theory Appl.*, vol. 169, no. 3, pp. 1042–1068, Jun. 2016.

[35] COIN-OR. *Ipopt*. Accessed: Mar. 25, 2021. [Online]. Available: https://coin-or.github.io/Ipopt/

[36] *Gurobi*. Accessed: Mar. 25, 2021. [Online]. Available: http://www.gurobi.com/

[37] *Birkhoffdecomposition.jl*. Accessed: Mar. 25, 2021. [Online]. Available: https://github.com/vvalls/BirkhoffDecomposition.jl

[38] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, Jan. 2017.

[39] COIN-OR. *CLP*. Accessed: Mar. 25, 2021. [Online]. Available: https://projects.coin-or.org/Clp