Online and Offline Algorithms for Circuit Switch Scheduling

Roy Schwartz
Technion – Israel Institute of Technology, Haifa, Israel
schwartz@cs.technion.ac.il

Mohit Singh
Georgia Institute of Technology, Atlanta, GA, USA
mohit.singh@isye.gatech.edu

Sina Yazdanbod
Georgia Institute of Technology, Atlanta, GA, USA
syazdanbod@gatech.edu

Abstract

Motivated by the use of high speed circuit switches in large scale data centers, we consider the problem of *circuit switch scheduling*. In this problem we are given demands between pairs of servers and the goal is to schedule at every time step a matching between the servers while maximizing the total satisfied demand over time. The crux of this scheduling problem is that once one shifts from one matching to a different one a fixed delay δ is incurred during which no data can be transmitted.

For the offline version of the problem we present a $(1 - 1/e - \epsilon)$ approximation ratio (for any constant $\epsilon > 0$). Since the natural linear programming relaxation for the problem has an unbounded integrality gap, we adopt a hybrid approach that combines the combinatorial greedy with randomized rounding of a different suitable linear program. For the online version of the problem we present a (bi-criteria) $((e-1)/(2e-1) - \epsilon)$-competitive ratio (for any constant $\epsilon > 0$) that exceeds time by an additive factor of $O(\delta/\epsilon)$. We note that no uni-criteria online algorithm is possible. Surprisingly, we obtain the result by reducing the online version to the offline one.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Scheduling algorithms

Keywords and phrases approximation algorithm, online, matching, scheduling

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.27

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.02800`.

Funding *Roy Schwartz*: Research is supported by NSF-BSF grant 2016742.
Mohit Singh: Research is supported by NSF- AF:1910423 and NSF-AF:1717947.

1 Introduction

In recent years the vast scaling up of data centers is fueled by applications such as cloud computing and large-scale data analytics. Such computational tasks, which are performed in a data center, are distributed in nature and are spread over thousands of servers. Thus, it is no surprise that designing better and efficient switching algorithms is a key ingredient in obtaining better use of networking resources. Recently, several works have focused on high speed optical circuit switches that have moving optical mirrors [6, 10, 28] or wireless circuits [13, 15, 29].

A common feature of many of these new switching models is that at any time the data can be transmitted on any matching between the senders and the receivers. However, once the switching algorithm decides to reconfigure from the current matching to a new different matching. This is due to physical limitations such as the time it takes to rotate mirrors, a fixed delay is incurred. This delay happens before data can be sent along the new reconfigured

matching. Indeed even if one mirror rotates the delay must be incurred and no data can be sent. This has led to significant study on obtaining good scheduling algorithms that take this delay into account [18, 21, 27]. The cost in switching between matchings makes the problem different when compared to the classical literature on scheduling in crossbar switching [5], which are usually based on Birkhoff von-Neumann decompositions. In this paper, we focus on finding the schedule that sends as much data as possible in a fixed time window. We aim to design simple and efficient offline and online algorithms, with provable guarantees, for the scheduling problem that incorporates switching delays.

In the circuit switch scheduling problem, we are given a traffic demand matrix $D \in \mathbb{R}_+^{|A|\times|B|}$, where A is the set of senders and B is the set of receivers. D_{ij} denotes the amount of data that needs to be sent from sender i to receiver j. The D_{ij}'s can also be seen as weights on the edges of a complete bipartite graph with vertex set $A \cup B$. We are also given a time window W and a switching time $\delta > 0$. At any time, the algorithm must pick a matching M and duration α for which the data is transmitted along the edges of the matching M that still require data to be sent. When the algorithm changes to another matching M' for another duration α', the algorithm must account for δ amount of time for switching between the two matchings. Indeed even if one edge changes in the matching, the delay must be incurred and no data can be sent on any of the matching edges. The total amount of time that data is sent along matchings as well as switching time between the matchings must total no more than W. The objective is to maximize the total demand that is satisfied.

1.1 Our Results and Contributions

Our main contribution in this paper are simple and efficient algorithms for the offline and online variants of the circuit switch scheduling problem. The following theorem summarizes our result for the offline setting which gives the first constant factor approximation algorithm for all instances.

▶ **Theorem 1.** *Given any constant $\epsilon > 0$, there is a polynomial time algorithm that returns a $(1 - 1/e - \epsilon)$-approximation for the circuit switch scheduling problem.*

It was already noted in Bojja et al. [27] that the circuit switch scheduling problem is a special case of maximizing a monotone submodular function given a knapsack constraint. Unfortunately, the above reduction requires a ground set of exponential size where elements in the ground set corresponds to matchings of senders and receivers. Hence, the rich literature on submodular function maximization (such as [24]) cannot be applied. Indeed the main challenge is the presence of exponential number of matchings that define the configurations.

Bojja et al. [27] show that the greedy algorithm can be implemented in polynomial time and give a guarantee under the assumption that all entries of the data matrix are small as compared to the time window. Unfortunately, it is easy to construct examples where the greedy algorithm does not give a guarantee close to $(1 - \frac{1}{e})$ (Refer to the full version of our paper [23] for an example).

A different approach is to formulate a linear programming relaxation and round the fractional solution. Indeed, it is easy to formulate two natural linear programming relaxations to the circuit switch scheduling problem. The first assigns a distribution over matchings for every time, whereas the second picks configurations with the additional knapsack constraint. Unfortunately, both have an unbounded integrality gap (Refer to the full version of our paper [23] for the gap examples). Thus, a different approach must be used.

We adopt a hybrid approach that combines greedy and rounding of a special linear program to prove the above theorem. We first give an improved analysis of the greedy algorithm and show it gives a $(1 - \frac{1}{e} - \epsilon)$ approximation when $\delta < \epsilon \cdot W$. On the other hand,

when $\delta \geq \epsilon W$, the optimal solution only contains $\frac{1}{\epsilon}$ different matchings. While it is not possible to even guess these constant number of matchings in the solution, we can enumerate (approximately) the time these unknown matchings are scheduled. We then formulate an assignment linear program that assigns matchings to each of these guessed time slots. Then a simple randomized rounding gives us the desired approximation in this case.

We also consider the online variant of the problem where the data matrix is not known in advance but is revealed over time. We consider a discrete time process where at each time step, we receive a new data matrix that needs to be transmitted in addition to the traffic demand still left from all preceding time steps. Moreover, we can choose a matching to transmit data at any time step with the constraint that whenever we change the matching from the previous step, no data is transmitted for δ steps. Our main contribution is a reduction from the online variant to the offline variant. To the best of our knowledge, such reductions with a minor loss in the guarantee are seldomly found. This results in a bi-criteria algorithm since the online algorithm is allowed a slightly larger time window than the optimum. We remark that such a bi-criteria approximation is necessary and we refer the reader to the full version of our paper [23] for details. The following theorem summarizes the above.

▶ **Theorem 2.** *Given a β-approximation for the offline circuit switch scheduling problem and an integer $k \geq 3$, there exists an algorithm achieving a competitive ratio of $(1 - 2/k) \frac{\beta}{1+(1-2/k)\beta}$ for the online circuit switch scheduling problem which uses a time window of $W + k\delta$ as compared to a time window of W for the optimum.*

Combining Theorem 1 and Theorem 2, we have the following corollary.

▶ **Corollary 3.** *For any constant $\epsilon > 0$, there exists an algorithm achieving a competitive ratio of $\left(\frac{e-1}{2e-1} - \epsilon\right)$ for the online circuit switch scheduling problem which uses a time window of $W + O(\delta/\epsilon)$ as compared to a time window of W for the optimum.*

We note that the online algorithm in the above corollary runs in polynomial time. If one is not interested in the running time of the algorithm, but rather interested only in coping with an unknown future, then Theorem 2 gives an online algorithm whose competitive ratio is $(1/2 - \epsilon)$ for any arbitrarily small constant $\epsilon > 0$ (by assuming that the offline problem can be solved optimally, i.e., $\beta = 1$).

1.2 Related Work

Bojja et al. [27] were the first to formally introduce the offline variant of the circuit switch scheduling problem. They focused on the special case that all entries of the data matrix are significantly small, and analyzed the greedy algorithm. Though it is known that the greedy algorithm does not provide any worst-case approximation guarantee for the general case of maximizing a monotone submodular function given a knapsack constraint, [27] proved that in the special case of small demand values, where $D_{ij} \leq \epsilon W$ for all i, j they obtain a $\left(1 - \frac{1}{e^{1-\epsilon}}\right)$-approximation. To the best of our knowledge, our algorithm gives the best provable bound for the offline variant of the circuit switch scheduling problem. A different related variant of the problem is when data does not have to reach its destination in one step, i.e., data can go through several different servers until it reaches its destination [18, 21, 27].

A dual approach, given by Liu et al. [20], aims to minimize the total needed time to transmit the entire demand matrix. Since our algorithm aims to maximize the transmitted data in a time window of W, one can use our algorithm as a black box while optimizing over W. It was proven in [19] that the problem of minimizing the time needed to send all of the data is NP-Complete. Hence, we conclude that the circuit switch scheduling problem is also NP-Complete.

The problem of decomposing a demand matrix into matchings, i.e., the decomposition of a matrix into permutation matrices, was considered by [3, 8, 17, 22]. The special cases of zero delay [14] and infinite delay [25] have also been considered. Several related, but slightly different, settings include [7, 11, 26].

Regarding the theoretical problem of maximizing a monotone submodular function given a knapsack constraint, Sviridenko [24] (building upon the work of Khuller et al. [16]) presented a tight $(1 - 1/e)$-approximation algorithm. This tight algorithm enumerates over all subsets of elements of size at most three, and greedily extends each subset of size three, and returns the best solution found. Deviating from the above combinatorial approach of [16, 24], Badanidiyuru and Vondrák [2] and Ene and Nguyen [9] present algorithms that are based on an approach that extrapolates between continuous and discrete techniques. Unfortunately, as previously mentioned, none of the above algorithms can be directly applied to the circuit switch problem due to the size of the ground set.

The online version of the circuit switch scheduling problem has been considered from a queuing theory prospective, with delays [4] and without delays [12]. In these works, guarantees are proven under the assumption that the incoming traffic is from a known distribution or i.i.d. random variables. To the best of our knowledge, the online version has not been studied from a theoretical perspective.

2 Preliminaries

First, let us start with a formal description of the problem. We are given a complete bipartite graph $G = (A, B, E)$ where A and B are the sets of sending and receiving servers, a constant $\delta \geq 0$ and a time window $W \geq 0$. We are also given the traffic demand matrix of the graph, $D \in \mathbb{R}_+^{|A|\times|B|}$, where D_{ij} denotes the amount of data that needs to be sent from sender i to receiver j. The D_{ij}'s can be seen as weights on the edges of the complete bipartite graph. To simplify the notation, for an edge $e = (i, j)$ we abbreviate D_{ij} to D_e. Let $\mathcal{M}$ be the collection of all matchings in G.

▶ **Definition 4.** *The pair (M, α) is called a configuration if $M \in \mathcal{M}$ and $\alpha \in \mathbb{R}_+$.*

The term *scheduling* a configuration (M, α) means sending data via the matching M for a duration of time that equals α. For simplicity of presentation, we also interpret a matching M as a $\{0, 1\}^{|A|\times|B|}$ matrix where $e \in M$ if and only if the entry of edge e in M equals 1. Note that for any edge $e \in M$ the total data sent through e would be $\min(D_e, \alpha)$ and the total amount of data sent by the configuration would be $||\min(D, \alpha M)||_1 = \sum_{e\in M} \min(D_e, \alpha)$ (note that the minimum is taken element-wise). For simplicity of presentation we may use $||.||_1$ and $||.||$ interchangeably.

Switching from a configuration (M, α) to another (M', α') incurs a given constant delay δ, during which no transmission is done. Let $\mathcal{C}$ denote the collection of all possible configurations.

▶ **Definition 5.** *A schedule S of size k is a subset $S \subseteq \mathcal{C}$ such that $|S| = k$. We say that S requires a total time of $\sum_{(M,\alpha)\in S}(\alpha + \delta)$ to be scheduled.*

The total time of the schedule includes both the time for sending data with each configuration and the delay in switching between them. This brings us to the definition of a feasible schedule.

▶ **Definition 6.** *A schedule S is feasible if $\sum_{\alpha:(M,\alpha)\in S}(\alpha + \delta) \leq W$.*

In the offline setting, the goal is to find a feasible schedule S that maximizes the data sent over the given time window of length W. This problem can be formulated as follows:

$$\max\left\{\left|\left|\min\left(D, \sum_{(M,\alpha)\in S}\alpha M\right)\right|\right|_1 : S \subseteq \mathcal{C}, \sum_{\alpha:(M,\alpha)\in S}(\alpha + \delta) \leq W\right\}. \tag{1}$$

We note that $\mathcal{C}$ might be of infinite size. However, we use standard discretization techniques to limit the set of possible values of α in our algorithms. We will discuss this with more detail in the later relevant sections. For now, assume $\mathcal{C}$ is finite. To facilitate the notation and the analysis of our problem, we turn to a well-known class of functions called *submodular functions*.

▶ **Definition 7.** *Given a ground set $N = \{1, 2, 3, ..., n\}$, a set function $f : 2^N \to \mathbb{R}_+$ is a submodular function if for every $A, B \subseteq N$: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.*

For our problem, define $f : 2^{\mathcal{C}} \to \mathbb{R}_+$ as:

$$f(S) = \left|\left|\min\left(D, \sum_{(M,\alpha)\in S} \alpha M\right)\right|\right|_1.$$

Moreover, we denote by $f_S((M,\alpha)) = f(S \cup (M,\alpha)) - f(S)$ the marginal gain of the schedule S if the configuration (M,α) was added to it. It has been shown that f is submodular (refer to Theorem 1 in [27]). For the sake of completeness, we state the theorem. Note that f is *monotone* if for every $A \subseteq B \subseteq N$: $f(A) \leq f(B)$.

▶ **Theorem 8** (Theorem 1 in [27]). *The function f is a monotone submodular function.*

For the online version of the problem, we use a discrete time model. Unlike the offline version, in the online setting we do not know the entire traffic matrix of the graph in the beginning. We start with D_0 as the demand matrix already present in the initial graph. At time t an additional traffic matrix D_t is revealed to the algorithm that includes new demands for data that need to be transmitted. In the online version of the problem sending configuration (M,α) means that for the next $\alpha \in \mathbb{Z}_+$ time steps our algorithm is busy sending the matching M. Switching a configuration to a different one incurs an additional delay of $\delta \in \mathbb{N}$ steps, during which no data can be sent. The incoming traffic matrices, at every step starting with the sending of (M,α) and ending with the switching cost (a total of $\alpha + \delta$ time steps), will accumulate and be added to the remaining traffic matrix of the graph.

3 Offline Circuit Switch Scheduling Problem

In this section, we prove Theorem 1 by giving an approximation algorithm for the circuit switch scheduling problem. Our algorithm is a combination of the greedy algorithm as well as a linear programming based approach. We first show that the greedy algorithm gives close to a $(1 - \frac{1}{e})$-approximation if δ, the switching time, is much smaller than the time window. This is done in Section 3.1. In Section 3.2, we give a randomized rounding algorithm for a linear programming relaxation that gives a $(1 - \frac{1}{e})$-approximation but runs in time exponential in number of matchings used in the optimal solution. While the natural linear program for the problem has unbounded gap, we show how to bypass this when the schedule has a constant number of matchings.

3.1 Greedy Algorithm

The greedy algorithm is as follows: at each step choose the configuration that maximizes the amount of data it sends per unit of time it uses. Formally, if R_i is the remaining data demand in the graph after i configurations were already chosen, the greedy algorithm will choose the following configuration to be used next:

$$(M_{i+1}, \alpha_{i+1}) = \text{argmax}_{M\in\mathcal{M},\alpha\in\mathbb{R}_+} \frac{||\min(R_i, \alpha M)||_1}{\alpha + \delta}. \tag{2}$$

Algorithm 1 Greedy Algorithm.

1: `Input:` $G = (A, B, E), D, \delta, W$
2: `Output:` $\{(M_1, \alpha_1), \ldots, (M_r, \alpha_r)\}$
3: $\mathcal{S} \leftarrow \emptyset$. $i \leftarrow 0$, $R_1 \leftarrow D$.
4: **while** $\sum_{\alpha:(M,\alpha)\in\mathcal{S}} (\alpha + \delta) \leq W$ **do**
5: $\quad i \leftarrow i + 1$, $(M_i, \alpha_i) \leftarrow \arg\max_{M\in\mathcal{M},\alpha\in\mathbb{R}_+} \frac{||\min(R_i, \alpha M)||}{\alpha+\delta}$.
6: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{(M_i, \alpha_i)\}$, $R_{i+1} \leftarrow R_i - \min(R_i, \alpha_i M_i)$.
7: **end while**
8: $r \leftarrow i$.
9: **if** $\sum_{(M,\alpha)\in\mathcal{S}} (\alpha + \delta) > W$ **then**
10: $\quad \beta_r \leftarrow W - \delta - \sum_{j=1}^{r-1} (\alpha_j + \delta)$
11: $\quad$ **if** $\beta_r \geq 0$ **then**
12: $\quad\quad \mathcal{S} \leftarrow (\mathcal{S} \setminus \{(M_r, \alpha_r)\}) \cup \{(M_r, \beta_r)\}$
13: $\quad$ **else**
14: $\quad\quad \mathcal{S} \leftarrow (\mathcal{S} \setminus \{(M_r, \alpha_r)\})$
15: $\quad$ **end if**
16: **end if**
17: **return** $\mathcal{S}$

The greedy algorithm continues to pick configurations until the first time the time constraint is violated or met. Algorithm 1 demonstrates this process. Let r denote this number of steps and $\mathcal{S}_r$ the schedule created after r steps of this algorithm. The last chosen configuration may violate the time window budget and a natural strategy is to reduce its duration to the time window W as is done in Step (11)-(12) of the algorithm. Indeed [27] analyzes this algorithm and shows that it performs well if each entry in data matrix is small. They also show that the above optimization problem can be solved using the maximum weight matching problem. We give a different analysis of the algorithm and show that it gives us a $\left(1 - \frac{1}{e} - \epsilon\right)$-approximation if $\delta < \left(\frac{e}{2(e-1)}\epsilon\right) \cdot W$.

▶ **Theorem 9.** *Let $\mathcal{S}_r$ denote the schedule as returned by the greedy algorithm and $\mathcal{O}$ denote the optimal schedule. Then*

$$f(\mathcal{S}_r) \geq \left(1 - \frac{2\delta}{W}\right)\left(1 - \frac{1}{e}\right) f(\mathcal{O}).$$

Proof. To analyze the algorithm, we first show that the objective of the optimal schedule of a slightly smaller time window $W - \delta$ is not much smaller than the optimum value of the optimum schedule for time window W in Lemma 10. Indeed, the lemma states that given any schedule for time window W, for example the optimal schedule, there exists a schedule with time window $W - \delta$ of a comparable objective.

▶ **Lemma 10.** *For any schedule $\mathcal{S}$ for a time window of W, there is a schedule $\tilde{\mathcal{S}}$ on a window of $W - \delta$ time such that $f(\tilde{\mathcal{S}}) \geq \left(1 - \frac{2\delta}{W}\right) f(\mathcal{S})$.*

Proof. Let T_{data} be the total time spent sending data and T_{switch} be the total time spent switching between configurations. Thus, $W = T_{\text{data}} + T_{\text{switch}}$. We prove that we can remove δ time from some configuration or we can remove an entire configuration from $\mathcal{S}$ while reducing the objective by no more than $\frac{2\delta}{W}$ fraction of the objective. Consider the two following cases for the given $\mathcal{S}$. If $T_{\text{data}} \geq \frac{W}{2}$, we have $\frac{f(\mathcal{S})}{T_{\text{data}}} \leq \frac{2}{W} f(\mathcal{S})$. Thus there exists a configuration

that we can deduct δ time from and at most lose $\frac{2\delta}{W} f(\mathcal{S})$. If $T_{\text{switch}} \geq \frac{W}{2}$. This means the number of configurations is at least $\frac{W}{2\delta}$. Each configuration on average sends $\frac{2\delta}{W} f(\mathcal{S})$ data. Therefore, there is a configuration we can completely remove from our schedule such that total amount of lost data is at most $\frac{2\delta}{W} f(\mathcal{S})$. In both cases we can reduce the time taken by the schedule by at least δ and have a new schedule $\tilde{\mathcal{S}}$ such that $f(\tilde{\mathcal{S}}) \geq \left(1 - \frac{2\delta}{W}\right) f(\mathcal{S})$. ◀

Let $\mathcal{O}'$ denote the optimal solution with time window $W - \delta$. From Lemma 10, we have $f(\mathcal{O}') \geq \left(1 - \frac{2\delta}{W}\right) f(\mathcal{O})$. In the following lemma, we show that the output of the greedy algorithm is at least a $\left(1 - \frac{1}{e}\right)$-approximation of $f(\mathcal{O}')$. The proof of the lemma follows standard analysis for greedy algorithms for coverage functions, or more generally submodular functions, except at the last step. For the proof refer to the full version of our paper [23]. The proof of Theorem 9 now follows immediately.

▶ **Lemma 11.** *If $\mathcal{O}'$ is the optimum schedule on time window $W - \delta$, then*

$$f(\mathcal{S}_r) \geq (1 - \frac{1}{e}) f(\mathcal{O}').$$ ◀

3.2 Linear Programming Approach for Constant Number of Configurations

In this section, we assume that we want to schedule at most a given constant k number of configurations and prove the following theorem.

▶ **Theorem 12.** *There exists a randomized polynomial time algorithm that given an integer k and an instance of the circuit switch scheduling problem returns a feasible schedule whose objective, in expectation, is at least $(1 - \frac{1}{e} - \epsilon)$ of the optimum solution that uses at most k matchings. Moreover the running time of the algorithm is polynomial in $\frac{n}{\epsilon^k}$.*

Let us denote optimum schedule by $\mathcal{O} = \{(M_1^*, \alpha_1^*), \ldots, (M_k^*, \alpha_k^*)\}$. Note that, without the loss of generality, we can assume that we know what the α_i^*'s are. This can be done by a standard discretization of the possible values. Since, the number of configurations is constant this enumeration will be polynomial in $\frac{1}{\epsilon^k}$ to an accuracy of ϵ. The total data sent by a schedule $\mathcal{S}$ is $f(\mathcal{S}) = ||\min(D, \sum_{(M,\alpha)\in\mathcal{S}} \alpha M)||_1$. However, in this section, it is more beneficial to consider the total data as the sum of total data sent over each edge. We model the total data by $Z = \sum_{e\in E} z_e$, where z_e is the amount of data that was sent through edge e in our graph. In the case of the optimum, $z_e^* = \min(D_e, \sum_{\alpha^*:(M^*,\alpha^*)\in\mathcal{O}:e\in M^*} \alpha^*)$ and $Z^* = \sum_{e\in E} z_e^*$. We can formulate the following integer program for this problem.[1]

$$
\begin{aligned}
(\mathcal{P}) \quad \max \quad & \sum_{e\in E} z_e && && (3)\\
s.t. \quad & \sum_{M\in\mathcal{M}} x_{M,i} \leq 1 && \forall i = 1,\ldots,k && (4)\\
& z_e \leq D_e && \forall e \in E && (5)\\
& z_e \leq \sum_{i=1}^{k} \sum_{M\in\mathcal{M}:e\in M} \alpha_i^* \cdot x_{M,i} && \forall e \in E && (6)\\
& x_{M,i} \in \{0,1\} && \forall e \in E, \forall M \in \mathcal{M}, \forall i = 1,\ldots,k &&
\end{aligned}
$$

[1] The variable $x_{M,i}$ is the fractional indicator for choosing the configuration (M, α_i^*).

Constraints (4) is to ensure that only one matching is considered in every time interval. Constraint (5) and (6) are to model the total data sent. We can relax this integer program to an LP by changing the $x_{M,i} \in \{0,1\}$ to $0 \leq x_{M,i} \leq 1$. The following lemma states that the relaxed linear program is a relaxation of our problem for the constant number of configurations.

▶ **Lemma 13.** *Let Z_{LP} be the value of an optimum solution to the LP, then $Z_{LP} \geq Z^*$*

Proof. If $\mathcal{O} = \{(M_1^*, \alpha_1^*), \ldots, (M_k^*, \alpha_k^*)\}$ is our optimum answer, based on $\mathcal{O}$ we will create a feasible answer to the LP. For every $(M_i^*, \alpha_i^*) \in \mathcal{O}$, we set $x_{M^*,i} = 1$. Clearly, the constraint 4 is satisfied since we picked exactly one matching for every interval. The constraints 5 and 6 are by definition satisfied since $f(\mathcal{O}) = ||\min(D, \sum_{(M,\alpha)\in\mathcal{O}} \alpha M)||$ and the constraints are modeling this minimum. This argument shows that the optimum answer is feasible in the LP and since the LP is a maximization problem we can conclude that $Z_{LP} \geq f(\mathcal{O})$. ◀

The LP contains an exponential number of variables, since the number of matchings in the complete graph is exponential in the size of the graph. To be able to solve this program we need to introduce a separation oracle for the dual of this LP. The following program is the dual of our LP.

$$(\mathcal{D}) \quad \min \quad \sum_{i=1}^{k} y_i + \sum_{e\in E} d_e a_e \tag{7}$$

$$s.t. \quad y_i \geq \alpha_i^* \sum_{e\in M} b_e \qquad \forall M \in \mathcal{M}, \forall i = 1, \ldots, k \tag{8}$$

$$a_e + b_e \geq 1 \qquad \forall e \in E \tag{9}$$

$$a_e \geq 0,\ b_e \geq 0,\ y_i \geq 0 \qquad \forall e \in E, \forall i = 1, \ldots, k$$

The Lemma 14 states the existence of a separation oracle.

▶ **Lemma 14.** *The dual program $\mathcal{D}$ admits a polynomial time separation oracle.*

Proof. Given a solution $(\{y_i\}_{i=1}^k, \{a_E\}_{e\in E}, \{b_e\}_{e\in E})$ we are required to determine whether it is feasible and if not provide a constraint that is violated. We can easily determine whether all constraints of type (9) are satisfied, and if not provide one that is violated, by a simple enumeration over all edges $e \in E$. The same can be done for constraints of type (8) by enumerating over $i = 1, \ldots, k$ and for each i compute a maximum weight matching in G equipped with $\{b_e\}_{e\in E}$ as edge weights and check whether the maximum weight matching has value at most y_i/α_i^*. If the maximum weight matching exceeds the target value return the constraint that corresponds to i and the maximum weight matching. ◀

Solving the linear program will provide us with a fractional solution $\{x_{M,i}\}_{M\in\mathcal{M},i=1,\ldots,k}$. For any i we have $\sum_{M\in\mathcal{M}} x_{M,i} \leq 1$. This constraint of the LP creates a distribution over the matchings in time interval i. We create a solution to the program $\mathcal{P}$ from the fractional solution by a randomized rounding technique. We pick $M \in \mathcal{M}$ for the time interval i with probability $x_{M,i}$. Note that with probability $1 - \sum_{M\in\mathcal{M}} x_{M,i}$ no matching will be chosen for this time interval. A formal description of this rounding method is provided in Algorithm 2. Let $X_{M,i}$ denote the indicator random variable if matching M is selected for the i^{th} slot. Moreover, let $Y_{e,i}$ denote the random variable that edge e is present in the matching chosen in the i^{th} slot. We have $Y_{e,i} = \sum_{M\in\mathcal{M}:e\in M} X_{M,i}$ for each $e \in E$ and i and $E[Y_{e,i}] = \sum_{M\in\mathcal{M}:e\in M} x_{M,i}$. Moreover, let Z_e denote the random variable that denotes the data sent along edge e. Then we have $Z_e = \min(D_e, \sum_{i=1}^k \alpha_i^* Y_{e,i})$. Observe that the random variables $\{Y_{e,i}\}_{i=1}^k$ are independent.

Algorithm 2 Randomized Rounding.

Input: $(k, \{\alpha_i^*\}_{i=1}^k, \{x_{M,i}\}_{M\in\mathcal{M},i=1,\ldots,k})$
Output: $\{(M_i, \alpha_i^*)\}_{i=1}^k$
for $i \leftarrow 1, \ldots, k$ **do**
    choose M_i to be a random matching w.p. $x_{M,i}$ for the interval i
end for
return $\{(M_i, \alpha_i^*)\}_{i=1}^k$

The following Lemma 15 is implicit in Theorem 4 of Andelman and Mansour [1].

▶ **Lemma 15.** *Let $Y_1, \ldots, Y_n$ be independent Bernoulli random variables and let $Z = \min(B, \sum_{i=1}^n b_i Y_i)$ for some non-negative reals $B, b_1, \ldots, b_n$. Then we have that $\mathbb{E}[Z] \geq \left(1 - \frac{1}{e}\right) \min\left(B, \mathbb{E}\left[\sum_{i=1}^n b_i Y_i\right]\right)$.*

Applying the above lemma for each e and random variables $\{Y_{e,i}\}_{i=1}^k$, we obtain that

$$\begin{aligned}\mathbb{E}[Z_e] &\geq \left(1-\frac{1}{e}\right)\min\left(D_e, \mathbb{E}\left[\sum_{i=1}^k \alpha_i^* Y_{e,i}\right]\right) = \left(1-\frac{1}{e}\right)\min\left(D_e, \sum_{i=1}^k \sum_{M\in\mathcal{M}:e\in M} \alpha_i^* x_{e,i}\right)\\ &\geq \left(1-\frac{1}{e}\right)\cdot z_e.\end{aligned}$$

Now summing over all edges, Theorem 12 follows. We are now ready to conclude our discussion of the offline variant of the circuit switch scheduling problem and prove Theorem 1.

Proof of Theorem 1. Given $\epsilon > 0$, if $\delta \leq (\frac{e}{2(e-1)}\epsilon)W$ then Theorem 9 gives us a $(1 - \frac{1}{e} - \epsilon)$-approximation. Otherwise, $\frac{2(e-1)}{e}\frac{1}{\epsilon} > \frac{W}{\delta}$ implying that at most $\frac{2(e-1)}{e}\frac{1}{\epsilon}$ configurations can be scheduled. In this case, Theorem 12 will give a $(1 - \frac{1}{e} - \epsilon)$-approximation. ◀

4 Online Circuit Switch Scheduling Problem

In this section, we prove Theorem 2. Recall that in the online setting, we consider a discrete time model[2] where an additional traffic matrix is revealed at every time $t = 1, 2, \ldots, T$. At every time step t, a new set of traffic demands arrives and adds to the remaining traffic that has not been sent so far. We assume that the data matrix arriving at each step is integral and thus can be modeled as a multigraph. We denote the incoming traffic matrices as multigraphs $\{E_1, E_2, \ldots, E_T\}$ (instead of D_i's to simplify and familiarize the notation) and thus union of any two such graphs is defined by adding the number of copies of edges in the two constituents. Before proving the general theorem, we first consider the case when there is no delay while switching matchings, i.e., $\delta = 0$. Observe that in this case, the offline problem can be solved exactly and we show a $\frac{1}{2}$-competitive algorithm for the online problem. The general reduction builds on this simple case along with the offline algorithm.

[2] We could also consider a continuous time model where data matrices can arrive at any time and the algorithm can choose a matching at any time instant with a switching time δ when no data is sent. Our results apply to this model as well. The discrete model makes the presentation of the results easier.

4.1 Without Configuration Delay

Observe that an online algorithm, in this case, will pick a set of matchings $\{M_1, M_2, \ldots, M_T\}$, instead of a schedule, that covers the maximum number of edges. At each step t, the algorithm picks the maximum matching from the graph formed by the new edges that arrive, E_t, and the remaining edges in the graph from previous steps which we denote by R_{t-1}. The algorithm is formally given in Algorithm 3. Here $\mathcal{M}$ denotes the set of all matchings on the complete bipartite graph with parts A and B. The objective of Algorithm 3 is $\sum_{t=1}^{T} |M_t|$, where $|M_t|$ denotes the number of edges in the matching M_t. We denote the optimum solution by $\mathcal{O} = \{O_1, \ldots, O_T\}$, We have the Theorem 16 for our approximation guarantee.

Algorithm 3 Online Greedy Algorithm without Delay.

1: `Input:` Bipartite multigraphs on $E_1, E_2, \ldots, E_T$ on $A \cup B$ where E_t is disclosed at beginning of step t.
2: `Output:` $\{M_1, M_2, \ldots, M_T\}$
3: $R_0, S \leftarrow \emptyset$, $t \leftarrow 1$.
4: **for** $t \leftarrow 1, 2, \ldots, T$ **do**
5: $\quad R'_t \leftarrow R_{t-1} \cup E_t$, $M_t \leftarrow argmax_{M \in \mathcal{M}, M \subseteq R'_t} |M|$.
6: $\quad S \leftarrow S \cup \{M_t\}$, $R_t \leftarrow R'_t \setminus \{M_t\}$, $t \leftarrow t + 1$.
7: **end for**
8: **return** S

▶ **Theorem 16.** *Algorithm 3 is $\frac{1}{2}$-competitive for the online circuit switch scheduling problem without delays.*

Proof. Let $\Gamma = \{E_1, \ldots, E_T\}$ denote the incoming edges for the first T steps. We call this the input sequence for the first T steps. We use induction on T to prove the theorem. Specifically, we prove that for *any* input sequence of edges for T steps, $\Gamma = \{E_1, E_2, \ldots, E_T\}$, we have $\sum_{t=1}^{T} |M_t| \geq \frac{1}{2} \sum_{t=1}^{T} |O_t|$.

For $T = 1$, we know that the maximum matching has the biggest size of any matching in the graph. So, we have $|M_1| \geq |O_1|$ and thus the base case holds. By the induction hypothesis, we have that for *any* input sequence of $T-1$ steps, we have $\sum_{t=1}^{T-1} |M_t| \geq \frac{1}{2} \sum_{t=1}^{T-1} |O_t|$ where $\{M_t\}_{t=1}^{T-1}$ and $\{O_t\}_{t=1}^{T-1}$ are the output of the algorithm and the optimal solution, respectively.

Now, consider any input sequence $E_1, \ldots, E_T$. Recall, R_1 is the residual graph formed after first step of the algorithm, i.e. $R_1 = E_1 \setminus M_1$. At the next step, the algorithm will find the maximum matching in $R'_2 = R_1 \cup E_2$ as its edge set. We build a new sequence of $T-1$ inputs and apply induction to it.

Let $\Gamma' = \{R'_2, E_3, \ldots, E_T\}$. Consider the optimum solution on this new input sequence. Let $\{M'_t\}_{t=2}^{T}$ be the matchings that our algorithm picks given this new input sequence and $\{O'_t\}_{t=2}^{T}$ the optimum matchings. Using the induction hypothesis we can write $\sum_{t=2}^{T} |M'_t| \geq \frac{1}{2} \sum_{t=2}^{T} |O'_t|$.

First note that for $2 \leq i \leq n$, $M_i = M'_i$. This is true since M_i and M'_i are the maximum matchings of the same graph as can be seen inductively. We now show the following lemma that relates the optimum solution of the new instance to the original instance.

▶ **Lemma 17.** $\sum_{t=2}^{T} |O'_t| \geq \sum_{t=2}^{T} |O_t| - |M_1|$.

Proof. The matchings $\{O_2 \setminus M_1, O_3 \setminus M_1, \ldots, O_T \setminus M_1\}$ is a feasible output for the optimum solution on the Γ' sequence. Therefore, we have $\sum_{t=2}^{T} |O'_t| \geq \sum_{t=2}^{T} |O_t| - |M_1|$ as required. ◀

Using the induction hypothesis and the lemma we can write

$$\sum_{t=2}^{T} |M_t| \geq \frac{1}{2}\left(\sum_{t=2}^{T} |O_t| - |M_1|\right)$$

Adding the inequality $|M_1| \geq |O_1|$ to both sides, we obtain

$$\sum_{t=1}^{T} |M_t| \geq \frac{1}{2}\left(\sum_{t=2}^{T} |O_t|\right) + \frac{1}{2}|M_1| \geq \frac{1}{2}\left(\sum_{t=2}^{T} |O_t|\right) + \frac{1}{2}|O_1| = \frac{1}{2}\left(\sum_{t=1}^{T} |O_t|\right)$$

and the induction step follows. ◀

4.2 With Configuration Delay

In this section, we assume switching between the configurations causes a delay of $\delta \in \mathbb{N}$ steps during which no data is sent. We also assume that we have access to a β-approximation for the offline version of the problem. Note that we view the offline algorithm as a black-box. More formally, we assume we have an algorithm of the form Algorithm 4. To reiterate, G is the given complete bipartite graph, D is the traffic demand matrix, δ is the switching delay and W is the size of the time window. Recall, that sending the configuration (M, α) means that for the next α steps we will only send data using matching M.

Algorithm 4 Offline Algorithm for Circuit Switch Scheduling.

1: `Input:` $G = (A, B, E), D, \delta, W$
2: `Output:` $\mathcal{S} = \{(M_1, \alpha_1), \ldots, (M_j, \alpha_j)\}$

Given a constant $k \geq 1$, the first step of the algorithm is to wait $k\delta$ steps for data to accumulate and then run the offline algorithm on the accumulated data for time window $W = k\delta$. Let $\mathcal{S}_1$ be the output of the offline algorithm. We run this schedule from time $t = k\delta + 1$ to $t = 2k\delta$. Meanwhile, we collect the incoming data matrices in these times. Figure 1 shows one step of the algorithm. At the next step, we consider the total remaining data that includes data that has not been scheduled so far from previous schedule(s) and newly arrived data in previous $k\delta$ steps. We then run the offline algorithm on this data matrix to obtain a schedule for the next $k\delta$ steps. More generally, we continue this process for every block of $k\delta$ time steps. Algorithm 5 is the formal description of the algorithm. Note that this description is written as an enumeration over blocks of size $k\delta$. Recall that $f(\mathcal{S})$ denotes the amount of data sent by any schedule $\mathcal{S}$.

Proof of Theorem 2. We use a coefficient $\gamma \leq \beta$ and optimize γ in the end. We prove the theorem by induction on the number of the blocks, i.e., l and will follow along the lines of proof of Theorem 16. As we did in the proof of Theorem 16, we consider the incoming traffic as sequences. But in this case we define a sequence $\Gamma = \{I_1, I_2, \ldots, I_l\}$, where $I_i = \bigcup_{j=(i-1)(k\delta)+1}^{i(k\delta)} D_j$ is the input of block i. For $l = 1$, let the optimum schedule be $\mathcal{O}$ and the algorithm's schedule be $\mathcal{S}$. Figure 1 shows this setting. Using Lemma 10, there exists a schedule $\tilde{\mathcal{O}}$ with the property that $f\left(\tilde{\mathcal{O}}\right) \geq \left(1 - \frac{2}{k}\right) f\left(\mathcal{O}\right)$. Since $\mathcal{S}$ is the output of our offline algorithm we can write $f\left(\mathcal{S}\right) \geq \beta f\left(\mathcal{O}'\right) \geq \left(1 - \frac{2}{k}\right) \beta f\left(\mathcal{O}\right) \geq \left(1 - \frac{2}{k}\right) \gamma f\left(\mathcal{O}\right)$ and the basis of the induction is proven.

For $l = t$, again let $\mathcal{O}$ be the optimum schedule and $\mathcal{S} = S_1 \cup S_2 \cdots \cup S_t$ be the output of our algorithm where each S_i is the schedule on ith $k\delta$ block. Let O_1 be the optimum schedule for the first block and S_1 our algorithm's schedule on that block. Refer to Figure 2 for an illustration of this setting.

Algorithm 5 Online Greedy with Delay.

```
Input: δ, k and data matrices D_1, D_2, ..., D_T on A × B where D_i revealed at beginning of step i. Let l = ⌈T/(kδ)⌉.
Output: S = S_1 ∪ S_2 ∪ ... ∪ S_l.
S ← ∅, R_0 ← ∅.
for r ← 0, ..., l − 1 do
    R'_r ← R_r + Σ_{rkδ+1 ≤ j ≤ (r+1)kδ} D_j.
    S_r ← OfflineAlgorithm(G, R'_r, δ, kδ).
    R_{r+1} ← R'_r − min(R'_r, Σ_{(α,M)∈S_r} αM), S ← S ∪ S_r.
end for
return S
```

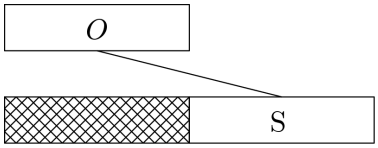

Figure 1 Basis of the induction. The crossed out block is the waiting period of our algorithm.

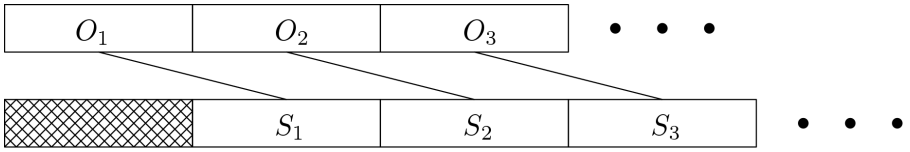

Figure 2 Step of the induction.

Consider the new input sequence $\Gamma' = \{R'_1 \cup I_2, I_3, \ldots, I_t\}$. Let the optimum schedule on the new input sequence be $\mathcal{O}'$ and the algorithm's schedule be $\mathcal{S}' = S'_1 \cup \cdots \cup S'_l$. From the induction hypothesis, we have $f(\mathcal{S}') \geq \left(1 - \frac{2}{k}\right) \gamma f(\mathcal{O}')$. Note that $S_i = S'_{i-1}$ for $i \geq 2$ and thus $f(\mathcal{S}') = f(\mathcal{S} \setminus S_1) = f(\mathcal{S}) - f(S_1)$. As in the proof of Lemma 17, a candidate schedule for the new instance is to consider $\mathcal{O} \setminus O_1$ and ignore the data sent by the algorithm in the schedule S_1 if it appears in any of the optimal matchings. Thus we obtain that

$$f(\mathcal{O}') \geq f(\mathcal{O} \setminus O_1) - f(S_1) = f(\mathcal{O}) - f(O_1) - f(S_1).$$

For O_1 based on our basis argument we can find S_1 such that $f(S_1) \geq \left(1 - \frac{2}{k}\right) \beta f(O_1)$. To sum up, we have the two following inequalities:

$$\begin{aligned} f(\mathcal{S}) - f(S_1) &\geq \left(1 - \frac{2}{k}\right) \gamma \left((f(\mathcal{O}) - f(O_1)) - f(S_1) \right), \\ f(S_1) &\geq \left(1 - \frac{2}{k}\right) \beta f(O_1). \end{aligned}$$

Rewriting the first inequality, we have

$$f(\mathcal{S}) - \left(1 - \left(1 - \frac{2}{k}\right)\gamma\right) f(S_1) \geq \left(1 - \frac{2}{k}\right) \gamma \left(f(\mathcal{O}) - f(O_1)\right)$$

Adding the $\left(1 - \left(1 - \frac{2}{k}\right)\gamma\right)$ times the second inequality

$$f(\mathcal{S}) \geq \left(1 - \frac{2}{k}\right) \gamma f(\mathcal{O}) - \left(1 - \frac{2}{k}\right) \left(\gamma - \beta \left(1 - \left(1 - \frac{2}{k}\right)\gamma\right)\right) f(O_1)$$

Optimizing the γ we get $\gamma = \frac{\beta}{\left(1 + \left(1 - \frac{2}{k}\right)\beta\right)}$ and thus proving the theorem. ◀

References

1 Nir Andelman and Yishay Mansour. Auctions with budget constraints. In *Scandinavian Workshop on Algorithm Theory*, pages 26–38, 2004.

2 Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1497–1514, 2014.

3 Siddharth Barman. Approximating Nash Equilibria and Dense Bipartite Subgraphs via an Approximate Version of Caratheodory's Theorem. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 361–369, 2015.

4 G Celik, Sem C Borst, Philip A Whiting, and Eytan Modiano. Dynamic scheduling with reconfiguration delays. *Queueing Systems*, 83(1-2):87–129, 2016.

5 Cheng-Shang Chang, Wen-Jyh Chen, and Hsiang-Yi Huang. On service guarantees for input-buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann. In *1999 Seventh International Workshop on Quality of Service (IWQoS'99)*, pages 79–86, 1999.

6 K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen. OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility. *IEEE/ACM Transactions on Networking*, 22(2):498–511, 2014.

7 Abel Dasylva and R Srikant. Optimal WDM schedules for optical star networks. *IEEE/ACM Transactions on Networking*, 7(3):446–456, 1999.

8 Fanny Dufossé, Kamer Kaya, Ioannis Panagiotas, and Bora Uçar. Further notes on Birkhoff–von Neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 554:68–78, 2018.

9 Alina Ene and Huy L. Nguyen. A Nearly-linear Time Algorithm for Submodular Maximization with a Knapsack Constraint. *CoRR*, abs/1709.09767, 2017. `arXiv:1709.09767`.

10 Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 40(4):339–350, 2010.

11 Shu Fu, Bin Wu, Xiaohong Jiang, Achille Pattavina, Lei Zhang, and Shizhong Xu. Cost and delay tradeoff in three-stage switch architecture for data center networks. In *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, pages 56–61, 2013.

12 Leonidas Georgiadis, Michael J Neely, Leandros Tassiulas, et al. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends® in Networking*, 1(1):1–144, 2006.

13 Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R Das, Jon P Longtin, Himanshu Shah, and Ashish Tanwer. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *ACM SIGCOMM Computer Communication Review*, volume 44 (4), pages 319–330, 2014.

14 Thomas Inukai. An efficient SS/TDMA time slot assignment algorithm. *IEEE Transactions on Communications*, 27(10):1449–1455, 1979.

15 Srikanth Kandula, Jitendra Padhye, and Victor Bahl. Flyways to DeCongest Data Center Networks. *Proc. of Hot Nets*, 2009.

16 Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999.

17 Janardhan Kulkarni, Euiwoong Lee, and Mohit Singh. Minimum Birkhoff-von Neumann Decomposition. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 343–354, 2017.

18 Conglong Li, Matthew K. Mukerjee, David G. Andersen, Srinivasan Seshan, Michael Kaminsky, George Porter, and Alex C. Snoeren. Using Indirect Routing to Recover from Network Traffic Scheduling Estimation Error. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, ANCS '17, pages 13–24, 2017.

19 Xin Li and Mounir Hamdi. On scheduling optical packet switches with reconfiguration delay. *IEEE Journal on Selected Areas in Communications*, 21(7):1156–1164, 2003.

20 He Liu, Matthew K Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M Voelker, David G Andersen, Michael Kaminsky, et al. Scheduling techniques for hybrid circuit/packet networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, page 41, 2015.

21 Liang Liu, Long Gong, Sen Yang, Jun Xu, and Lance Fortnow. Better Algorithms for Hybrid Circuit and Packet Switching in Data Centers. *arXiv preprint*, 2017. `arXiv:1712.06634`.

22 Vahab Mirrokni, Renato Paes Leme, Adrian Vladu, and Sam Chiu wai Wong. Tight Bounds for Approximate Carathéodory and Beyond. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2440–2448, 2017.

23 Roy Schwartz, Mohit Singh, and Sina Yazdanbod. Online and Offline Greedy Algorithms for Routing with Switching Costs. *arXiv preprint*, 2019. `arXiv:1905.02800`.

24 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

25 Brian Towles and William J Dally. Guaranteed scheduling for switches with configuration overhead. *IEEE/ACM Transactions on Networking*, 11(5):835–847, 2003.

26 Shay Vargaftik, Katherine Barabash, Yaniv Ben-Itzhak, Ofer Biran, Isaac Keslassy, Dean Lorenz, and Ariel Orda. Composite-path switching. In *Proceedings of the 12th International on Conference on emerging Networking Experiments and Technologies*, pages 329–343, 2016.

27 S. Bojja Venkatakrishnan, M. Alizadeh, and P. Viswanath. Costly circuits, submodular schedules and approximate carathéodory theorems. *Queueing Systems*, pages 1–37, 2018.

28 Guohui Wang, David G Andersen, Michael Kaminsky, Konstantina Papagiannaki, TS Ng, Michael Kozuch, and Michael Ryan. c-Through: Part-time optics in data centers. In *ACM SIGCOMM Computer Communication Review*, volume 40 (4), pages 327–338, 2010.

29 Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y Zhao, and Haitao Zheng. Mirror mirror on the ceiling: Flexible wireless links for data centers. *ACM SIGCOMM Computer Communication Review*, 42(4):443–454, 2012.