ONLINE AND OFFLINE ALGORITHMS FOR CIRCUIT SWITCH SCHEDULING

A Dissertation
Presented to
The Academic Faculty

By

Sina Yazdanbod

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
College of Computing

Georgia Institute of Technology

August 2020

ONLINE AND OFFLINE ALGORITHMS FOR CIRCUIT SWITCH SCHEDULING

Date Approved: May 11, 2020

First say to yourself what you would be; and then do what you have to do.

Epictetus

Dedicated to those who helped along the way.

ACKNOWLEDGEMENTS

I want to thank my thesis advisor Prof. Mohit Singh for their support throughout this project. Their guidance helped me grow as a researcher and their patience helped me grow as a person. I also would like to thank our collaborator Prof. Roy Schwartz for their help and thoughtful discussions. I am indebted to Prof. Jim Xu and Prof. Siva Theja Maguluri who agreed to take time our of their busy schedule during the COVID-19 crisis to be a part my committee.

I like to thank my friends and colleagues, Mehrdad, Bhuvesh, Saurabh, Samira, Tao, Zongchen, Majid, Matt and many others who helped me throughout this path.

Finally, my family that always supported me and helped in the pursue of this degree.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

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.

CHAPTER 1
INTRODUCTION AND OUR CONTRIBUTION

1.1 Motivation

In recent years the vast scaling up of data centers is fueled by applications such as cloud computing and large-scale data analytic. 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 [1, 2, 3] or wireless circuits [4, 5, 6].

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, due to physical limitations such as the time it takes to rotate mirrors, a fixed delay is incurred before data can be sent along the new reconfigured matching. This has led to significant study on obtaining good scheduling algorithms that take this delay into account [7, 8, 9]. The cost in switching between matchings makes the problem different when compared to the classical literature on scheduling in crossbar switching [10], 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. 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.

1.2 Preliminaries

In this section, we will define the basics of the definitions that we will need throughout this thesis. We will not provide full details of what these processes are but the facts and theorems that we will use about them. In the next section, we define our problem setting using these basic definitions. We introduce a greedy algorithm for monotone submodular function maximization and the ideas behind solving a linear program that has an exponential number of constraints.

1.2.1 Submodular Functions

In this section, we introduced a very useful class of functions called Submodular functions. There is a vast amount of literature on these functions for which we refer the reader to [11]. Given a ground set N, A submodular functions is a function $f : 2^N \to \mathbb{R}$ with the property that for $A, B \subseteq N$,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{1.1}$$

This property is called the submodular property. This property is equal to the following: For any $A \subseteq B \subseteq [n]$ and $i \in N\ B$, we have

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B) \tag{1.2}$$

It has been shown that a function is submodular if and only if either Property 1.1 and Property 1.2 holds. Property 1.2 is also called the property of diminishing returns. Since, as the set grows bigger the value gained by the growth is decreasing. A submodular function is monotone if we have $f(A) \leq f(B)$ for $A, B \subseteq N$ if and only if $A \subseteq B$. In this thesis we only consider the monotone submodular functions and we might refrain from specifying its monotonicity every time. A natural problem regrading the submodular functions is the following optimization problem.

$$\begin{aligned} \underset{S \subseteq N}{\text{maximize}} \quad & f(S) \\ \text{subject to} \quad & \sum_{s \in S} w(s) \leq k \end{aligned}$$

Where $k \in \mathbb{N}$ is called the budget and w is a positive weight function on the ground set. This problem is called Monotone Submodular Maximization subject to Knapsack Constraint. In general this problem is NP-Complete. For the exact details of how submodular functions work and proofs refer to [12].

1.2.2 Linear Programming

Linear programming, as we will use it, is the process of solving a system of linear equations. Consider the following program

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned}$$

In which, x is the vector of variables and c,b are the vectors of coefficients and A is a matrix. The systems of linear programs with polynomial constraints are always solvable in polynomial time [13]. However, if the number of constraints are exponential, we need extra assumptions to solve the linear program. One of the most common assumptions is the existence of a separation oracle. A separation oracle is a process that given x^*, it can decide whether x^* satisfies all constraints or gives at least one inequality such that $ax^* > b$. If such oracle is available, linear programs with exponential number of variables are solvable using the Ellipsoid Method. For more information on Ellipsoid Method, we refer the reader to [13].

1.3 Circuit Switch Problem Setting

In this section, we describe the problem formally. 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 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 1. *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 times 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 is 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 can be made. Let $\mathcal{C}$ denote the collection of all possible configurations.

Definition 2. *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 3. *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 W. This problem can be formulated as follows:

$$\max \left\{ \left|\left| \min \left(D, \sum_{(M,\alpha)\in S} \alpha M \right) \right|\right|_1 : S \subseteq \mathcal{C}, \sum_{\alpha:(M,\alpha)\in S} (\alpha + \delta) \leq W \right\} \tag{1.3}$$

We note that $\mathcal{C}$ might be of infinite size. However, we use standard discretization techniques to limit the set of possible values for 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 4. *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 [9]). 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 1 (Theorem 1 in [9])**.** *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.

1.4 Our 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.

Theorem 2. *Given any constant $\epsilon > 0$, there is an algorithm that returns a $(1 - {}^{1}\!/_{e} - \epsilon)$-approximation for the circuit switch scheduling problem.*

We note that two natural linear programming relaxations, to the problem both have an unbounded integrality gap. The first assigns a distribution over matchings for every time, and the second picks configurations with the additional knapsack constraint. 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. The former approach is employed when the switching delay δ is significantly smaller than the time window W, while the latter approach is employed otherwise. It was already noted [9] that the offline variant of 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 infinite size where each element in the ground set corresponds to a matching M and a duration α. We note that even if the duration are discretized we are still left with a ground set of exponential size that contains all matchings of the bipartite graph. Thus, the standard tight $(1 - 1/e)$-approximation [12] for maximizing a monotone submodular function given a knapsack constraint cannot be applied. Our main technical ingredient is to show that despite the above difficulties, the hybrid approach we propose in the paper allows one to obtain the nearly optimal $(1 - 1/e)$-approximation for the problem.

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 additional data matrix that needs to be transmitted. 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. This results in a bi-criteria algorithm since the online algorithm is allowed a slightly larger time window than the optimum.

Theorem 3. *Given a β-approximation for the offline circuit switch scheduling problem and an integer $k \geq 1$, there exists an algorithm achieving a competitive ratio of $\frac{(1-2/k)\beta}{1+(1-2/k)\beta}$ 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 2 and Theorem 3, we have the following corollary.

Corollary 3.1. *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 $W + O\left({}^{\delta}/_{\epsilon}\right)$ as compared to a time window of W for the optimum.*

1.5 Related Works

Venkatakrishnan et. al. [9] 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, [9] proved that in the special case of small demand values they obtain an (almost) tight approximation guarantee. 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 [7, 8, 9].

A dual approach is given by Liu et. al. [14], who aim 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 [15] that the problem of minimizing the time needed to send all of the data is NP-Complete. Hence, we can 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 [16, 17, 18, 19]. The special cases of zero delay [20] and infinite delay [21] have also been considered. Several related, but slightly different, settings include [22, 23, 24].

Regarding the theoretical problem of maximizing a monotone submodular function given a knapsack constraint, Sviridenko [12] (building upon the work of Khuller et. al. [25]) 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 [25, 12], Badanidiyuru and Vondrák [26] and Ene and Nguy˜ên [27] 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 has been considered from a queuing theory prospective. In both with delay[28] and without [29]. In which expectation guarantees are proven under the assumption that the incoming traffic is from a previously known distribution or i.i.d. random variables. To the best of our knowledge, the online version has not been studied from a theoretical prospective.

CHAPTER 2
OFFLINE CIRCUIT SWITCH ALGORITHM

In this section, we prove Theorem 2 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 approach. We first show that the greedy algorithm gives close to a $(1 - \frac{1}{e})$-approximation if δ, the switching delay, is much smaller than the time window W. Notice that this is the case where we have a large number of configurations. The other case, where δ is close to W, we are limited in the number of configurations we can use. In Section 2.2, 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. Our joint method uses both algorithms and picks the best approximation as the answer for the problem.

2.1 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:

$$(M_{i+1}, \alpha_{i+1}) = \text{argmax}_{M \in \mathcal{M}, \alpha \in \mathbb{R}_+} \frac{||\min(R_i, \alpha M)||_1}{\alpha + \delta}. \qquad (2.1)$$

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

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:     $i \leftarrow i + 1$.
6:     $(M_i, \alpha_i) \leftarrow \arg\max_{M\in\mathcal{M},\alpha\in\mathbb{R}_+} \frac{||\min(R_i,\alpha M)||}{\alpha+\delta}$.
7:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{(M_i, \alpha_i)\}$.
8:     $R_{i+1} \leftarrow R_i - \min(R_i, \alpha_i M_i)$.
9: end while
10: $r \leftarrow i$.
11: if $\sum_{(M,\alpha)\in\mathcal{S}} (\alpha + \delta) > W$ then
12:     $\beta_r \leftarrow W - \delta - \sum_{j=1}^{r-1}(\alpha_j + \delta)$
13:     if $\beta_r \geq 0$ then
14:         $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{(M_r, \alpha_r)\}) \cup \{(M_r, \beta)\}$
15:     else
16:         $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{(M_r, \alpha_r)\})$
17:     end if
18: end if
19: return $\mathcal{S}$
```

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 [9] 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 $\delta < (\frac{e}{2(e-1)}\epsilon) \cdot W$.

Theorem 4. *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 1. 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 1. *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 $f(\tilde{\mathcal{S}}) \geq \left(1-\frac{2\delta}{W}\right) f\left(\mathcal{S}\right)$.*

Proof. Let T_{data} be the total time spent sending data and T_{switch} be the total time spent switching between configurations. Thus, $W = T_{data} + T_{switch}$. 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 $T_{data} \geq \frac{W}{2}$. So we have $\frac{f(\mathcal{S})}{T_{data}} \leq \frac{2}{W} f\left(\mathcal{S}\right)$. This means, there exists a configuration that we can deduct δ time from and at most lose $\frac{2\delta}{W} f\left(\mathcal{S}\right)$. If $T_{switch} \geq \frac{W}{2}$. This means the number of configurations is at least $\frac{W}{2\delta}$. Each configuration on average sends $\frac{2\delta}{W} f\left(\mathcal{S}\right)$ data. Therefore, there is a configuration we can completely remove from our schedule such that total amount of lost data is at most $\frac{2\delta}{W} f\left(\mathcal{S}\right)$. 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 $f\left(\tilde{\mathcal{S}}\right) \geq \left(1-\frac{2\delta}{W}\right) f\left(\mathcal{S}\right)$. □

Let $\mathcal{O}'$ denote the optimal solution with time window $W-\delta$. From Lemma 1, 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 for the being careful at the last step.

Lemma 2. *If $\mathcal{O}'$ is the optimum schedule on time window $W-\delta$, then*

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

Proof. Let $\mathcal{S}'_r = \{(M_1,\alpha_1), \ldots, (M_r,\alpha_r)\}$ be the set of configurations picked by the greedy algorithm before the update steps (11)-(12) in which α_r is reduced to $\beta_r := W - \delta - \sum_{j=1}^{r-1}(\alpha_j + \delta)$ to obtain schedule $\mathcal{S}_r$. Note that β_r could be negative, however, for now assume $\beta_r \geq 0$. For ease of notation we also define $\beta_i = \alpha_i$ for each $1 \leq i \leq r-1$. Thus $\mathcal{S}_r = \{(M_1, \beta_1), \ldots, (M_r, \beta_r)\}$. We also let $\mathcal{S}_i$ to be the scheduled formed by picking the first i configurations in $\mathcal{S}_r$. We now show the following claim.

Claim 1. *For any configuration (M_i, β_i) picked by the greedy algorithm in schedule $\mathcal{S}_r$ at any $1 \leq i \leq r$, we have*

$$f_{\mathcal{S}_{i-1}}\left((M_i, \beta_i)\right) \geq \frac{\beta_i + \delta}{W - \delta}\left(f\left(\mathcal{O}'\right) - f\left(\mathcal{S}_{i-1}\right)\right).$$

Proof. First let us concentrate on the case when $i < r$. Then $\beta_i = \alpha_i$. Note that since M_i is a matching that maximizes $\frac{||\min(R_i, \alpha_i M_i)||}{\alpha_i + \delta}$, for any other $M \in \mathcal{M} \setminus \mathcal{S}_{i-1}$ and any $\alpha \in \mathbb{R}_+$ we can write

$$\frac{f_{\mathcal{S}_{i-1}}\left((M, \alpha)\right)}{\alpha + \delta} \leq \frac{f_{\mathcal{S}_{i-1}}\left((M_i, \alpha_i)\right)}{\alpha_i + \delta}$$

or equivalently, for each $1 \leq i \leq r$ and configuration (M, α), we have

$$f_{\mathcal{S}_{i-1}}\left((M, \alpha)\right) \leq \frac{\alpha + \delta}{\alpha_i + \delta} f_{\mathcal{S}_{i-1}}\left((M_i, \alpha_i)\right). \tag{2.2}$$

For any $1 \leq i \leq r$, consider the following

$$f\left(\mathcal{O}'\right) - f\left(\mathcal{S}_{i-1}\right) \leq f\left(\mathcal{O}' \cup \mathcal{S}_{i-1}\right) - f\left(\mathcal{S}_{i-1}\right) = f_{\mathcal{S}_{i-1}}\left(\mathcal{O}'\right) \leq \sum_{(M,\alpha) \in \mathcal{O}' \setminus \mathcal{S}_{i-1}} f_{\mathcal{S}_{i-1}}\left((M, \alpha)\right) \tag{2.3}$$

The last inequality comes from the submodularity of the function. Summing Inequality (2.2) over all configurations in $\mathcal{O}' \setminus \mathcal{S}_{i-1}$ and using that the $\mathcal{O}'$ has a time window $W - \delta$, we obtain that

$$\sum_{(M,\alpha) \in \mathcal{O}' \setminus \mathcal{S}_{i-1}} f_{\mathcal{S}_{i-1}}\left((M, \alpha)\right) \leq \frac{W - \delta}{\alpha_i + \delta} f_{\mathcal{S}_{i-1}}\left((M_i, \alpha_i)\right).$$

Combining the above inequality with Inequality (2.3), we obtain

$$f_{\mathcal{S}_{i-1}}\left((M_i, \alpha_i)\right) \geq \frac{\alpha_i + \delta}{W - \delta}\left(f\left(\mathcal{O}'\right) - f\left(\mathcal{S}_{i-1}\right)\right). \tag{2.4}$$

Thus if $i < r$, the claim follows since we have $\beta_i = \alpha_i$. When $i = r$, first observe that since the data sent along a single matching is a concave function of the time it is used in a configuration, we have that

$$\begin{aligned} f_{\mathcal{S}_{r-1}}\left((M_r, \beta_r)\right) &\geq \frac{\beta_r + \delta}{\alpha_r + \delta} f_{\mathcal{S}_{r-1}}\left((M_r, \alpha_r)\right) \\ &\geq \frac{\beta_r + \delta}{\alpha_r + \delta}\frac{\alpha_r + \delta}{W - \delta}\left(f\left(\mathcal{O}'\right) - f\left(\mathcal{S}_{r-1}\right)\right) \geq \frac{\beta_r + \delta}{W - \delta}\left(f\left(\mathcal{O}'\right) - f\left(\mathcal{S}_{r-1}\right)\right) \end{aligned}$$

This completes the proof of the claim. □

First note we can write the following equality:

$$f\left(\mathcal{O}\right) - f\left(\mathcal{S}_r\right) = f\left(\mathcal{O}\right) - f\left(\mathcal{S}_{r-1}\right) - f_{\mathcal{S}_{r-1}}\left((\mathcal{M}_r, \beta_r)\right)$$

We will now derive the approximation factor.

$$f\left(\mathcal{O}\right) - f\left(\mathcal{S}_{r-1}\right) - f_{\mathcal{S}_{r-1}}\left((\mathcal{M}_r, \beta_r)\right) \geq f\left(\mathcal{O}\right) - f\left(\mathcal{S}_{r-1}\right) - \frac{\beta_r + \delta}{W - \delta}\left(f\left(\mathcal{O}\right) - f\left(\mathcal{S}_{r-1}\right)\right)$$

This will result in the following inequality.

$$f\left(\mathcal{O}\right) - f\left(\mathcal{S}_r\right) \geq \left(f\left(\mathcal{O}\right) - f\left(\mathcal{S}_{r-1}\right)\right)\left(1 - \frac{\beta_r + \delta}{W - \delta}\right)$$

Continuing for the remaining $r - 1$ steps we will have:

$$f\left(\mathcal{O}\right) - f\left(\mathcal{S}_r\right) \geq \left(f\left(\mathcal{O}\right) - f\left(\mathcal{S}_0\right)\right)\Pi_{i=1}^{r}\left(1 - \frac{\beta_i + \delta}{W - \delta}\right)$$

Now using $1 - x \le e^{-x}$, we can write:

$$f(\mathcal{O}) - f(\mathcal{S}_r) \ge f(\mathcal{O})\, e^{-\sum_{i=1}^{r}\left(\frac{\beta_i+\delta}{W-\delta}\right)}$$

Since $\sum_{i=1}^{r}(\beta_i + \delta) \ge W - \delta$,

$$f(\mathcal{S}_r) \ge \left(1 - \frac{1}{e}\right) f(\mathcal{O})$$

Thus concluding the theorem for $\beta_r \ge 0$. Note that if $\beta_r < 0$, the whole argument of this section still holds without considering β_r and the last configuration. This is because $\sum_{i=1}^{r-1}(\beta_i + \delta) \ge W - \delta$ holds without the last configuration if $\beta_r < 0$. Notice that in this case the algorithm will drop the last configuration. $\square$

The proof of Theorem 4 now follows. $\square$

Here we take a brief moment to analyze the time of this algorithm. The main time of the Algorithm 1 is in how many times the loop will repeat and in each iteration how long line 6 will take. It is clear that the maximum number of configurations that we can schedule is $\frac{W}{\delta}$. Since, every configuration, regradless of α, will take at least δ units of time. This means the loop will repeat at most $\frac{W}{\delta}$ times. Now, solving the maximum in line 6 relies on a simple fact. The α that maximizes $\arg\max_{M\in\mathcal{M},\alpha\in\mathbb{R}_+} \frac{||\min(R_i,\alpha M)||}{\alpha+\delta}$ is an element of the remaining traffic matrix, R_i. This simplifies the search for α into a binary search on the non-zero elements remaining on the traffic matrix. For a proof of this fact we refer the reader to appendix of [9]. For every α in the process of the binary search we find the maximum weighted matching in the bipartite graph to determine $\frac{||\min(R_i,\alpha M)||}{\alpha+\delta}$. In conclusion, we will have an algorithm of order $O(d^2(\frac{n^5}{2})\frac{W}{\delta})$, where d is the density of the graph. For an exact break down of this refer to [9].

2.2 LP 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 5. *If $\delta < (\frac{e}{2(e-1)}\epsilon) \cdot W$, 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.*

Let us denote optimum schedule by $\mathcal{O} = \{(M_1^*, \alpha_1^*), \ldots, (M_k^*, \alpha_k^*)\}$ and let $D_{max} = \max_{e \in E} D_e$ where D_e is the data on edge e. let's scale W and all traffic matrix to numbers between 0 and 1. This can be done since if $D_{max} > W$, we know no edge can send data more than W in the entire schedule, so we can cut the extra data to have $D_{max} = W$. We can assume that we know what the α_i^*'s are. This can be done by a standard discretization of the possible values. Consider a $\lambda < 1$, We can estimate α_i^*'s to be $\{\lambda, 2\lambda, \ldots, 1\}$, we now briefly show that this estimation is polynomial and loses at most $\epsilon f(\mathcal{O})$ for the correct λ. Consider $\alpha_i^* = j\lambda$ where $j\lambda$ is the closest multiply of λ such that $j\lambda \leq \alpha_i^* < (j+1)\lambda$. In this case $f(\{(M_1^*, j\lambda)\}) \geq f(\{(M_1^*, \alpha_i^*)\}) - f(\{(M_1^*, \lambda)\}) \geq f(\{(M_1^*, \alpha_i^*)\}) - \lambda n$. This is because the data that M_1 can send in a unit of time is bounded by λn. Consider $O' = \{(M_1^*, \lambda_1^*), \ldots, (M_k^*, \lambda_k^*)\}$, where λ_i^* is the appropriate λ's we picked above. Note that $\sum_1^k \lambda_i^* \leq 1$. For the total data we have, $f(O') \geq f(\mathcal{O}) - nk\lambda$. Now we want $nk\lambda \leq \epsilon f(\mathcal{O}) \leq \epsilon n$. Meaning $\lambda = \frac{\epsilon}{k}$. The total number of different λ's that we need to try is at most $\frac{1}{\lambda}$ for every configuration and $\frac{1}{\lambda^k}$ total.

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.

$$
\begin{aligned}
(\mathcal{P}) \quad \max \quad & \sum_{e \in E} z_e && && (2.5)\\
s.t. \quad & \sum_{M \in \mathcal{M}} x_{M,i} \leq 1 && \forall i = 1, \ldots, k && (2.6)\\
& z_e \leq D_e && \forall e \in E && (2.7)\\
& z_e \leq \sum_{i=1}^{k} \sum_{M \in \mathcal{M}: e \in M} \alpha_i^* \cdot x_{M,i} && \forall e \in E && (2.8)\\
& x_{M,i} \in \{0, 1\} && \forall e \in E, \forall M \in \mathcal{M}, \forall i = 1, \ldots, k &&
\end{aligned}
$$

In $(\mathcal{P})$, z_e is a variable for edge e which shows the amount of data sent through the edge e using the entire schedule. Since we only need k configurations in this setting, $x_{M,i}$ is a variable showing partial participation of matching M in configuration i (Note that due to the nature of linear programming we have partial configurations here, where they include more than one matching). Constraints (2.6) is to ensure that only one matching is considered in every time interval. Constraint (2.7) and (2.8) 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 3. *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 2.6 is satisfied since we picked exactly one matching for every interval. The constraints 2.7 and 2.8 is 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 $Z_{LP} \geq f(\mathcal{O})$. $\square$

The LP contains an exponential number of variables, since the number of matchings in a graph could be exponential. To be able to solve this program we need to introduce a separation oracle for the dual of this LP. Existence of the separation oracle proves the linear program can be solved using the ellipsoid method. Using the ellipsoid on the dual will provide a polynomial set of violated dual variables that the separation oracle determines. We can replace the constraints of the original LP with only the set of the violated dual variables. This will create a polynomial size LP that is solvable in polynomial time. Thus using ellipsoid method on the dual will provide a way to find the primal variables. For the exact details of this process see Lemma 3 of [30]. The following program is the dual:

$$
\begin{aligned}
(\mathcal{D}) \quad \min \quad & \sum_{i=1}^{k} y_i + \sum_{e \in E} D_e a_e && && (2.9)\\
s.t. \quad & y_i \geq \alpha_i^* \sum_{e \in M} b_e && \forall M \in \mathcal{M}, \forall i = 1, \ldots, k && (2.10)\\
& a_e + b_e \geq 1 && \forall e \in E && (2.11)\\
& a_e \geq 0,\ b_e \geq 0,\ y_i \geq 0 && \forall e \in E, \forall i = 1, \ldots, k &&
\end{aligned}
$$

The Lemma 4 states the existence of a separation oracle.

Lemma 4. *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 (2.11) 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 (2.10) 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 $\{x_{M,i}\}_{M \in \mathcal{M}, i=1,\ldots,k}$.

Algorithm 2 Randomized Rounding

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

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.

The following Lemma 5 is implicit in Theorem 4 of Andelman and Mansour [31].

Lemma 5. *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

$$\mathbb{E}[Z] \geq \left(1 - \frac{1}{e}\right) \min\left(B, \mathbb{E}\left[\sum_{i=1}^n b_i Y_i\right]\right).$$

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

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

Now summing over all edges, we have $\mathbb{E}[Z] \geq (1-\frac{1}{e})Z \geq (1-\frac{1}{e}-\epsilon)f(\mathcal{O})$ and Theorem 5 follows. We are now ready to conclude our discussion of the offline variant of the circuit switch scheduling problem and prove Theorem 2.

Proof of Theorem 2. Given $\epsilon > 0$, if $\delta \leq (\frac{e}{2(e-1)}\epsilon)W$ then Theorem 4 gives us a $(1-\frac{1}{e}-\epsilon)$. 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 5 will give a $(1-\frac{1}{e}-\epsilon)$ approximation. $\square$

CHAPTER 3
ONLINE CIRCUIT SWITCHING ALGORITHM

In this section, we prove Theorem 3. Recall that in the online setting, we consider a discrete time model where an additional traffic matrix is revealed at every time $t = 1, 2, \dots, T$. 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 $\{E_1, E_2, \dots, E_T\}$ (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.

3.1 Without Configuration Delay

Observe that an online algorithm, in this case, will pick a set of matchings $\{M_1, M_2, \dots, M_T\}$, 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 $\mathcal{O} = \{O_1, \dots, O_T\}$, We have the Theorem 6 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:     $R'_t \leftarrow R_{t-1} \cup E_t$.
6:     $M_t \leftarrow argmax_{M \in \mathcal{M}, M \subseteq R'_t} |M|$.
7:     $S \leftarrow S \cup \{M_t\}$, $R_t \leftarrow R'_t \setminus \{M_t\}$, $t \leftarrow t + 1$.
8: end for
9: return $S$
```

Theorem 6. *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 6. $\sum_{t=2}^{T} |O'_t| \geq \sum_{t=2}^{T} |O_t| - |M_1|$.

Proof. The matchings $\{O_2 \backslash M_1, O_3 \backslash M_1, \ldots, O_T \backslash M_1\}$ 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. □

3.2 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 3.1 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 $k\delta$. For the sake of simplicity, let $f(\mathcal{S})$ be amount of data sent by any schedule $\mathcal{S}$.

Algorithm 5 Online Greedy with Delay

```
1: Input: $\delta, k$ and data matrices $D_1, D_2, \ldots, D_W$ on $A \times B$ where $D_i$ revealed at beginning of step $i$. Let $l = \lceil \frac{T}{k\delta} \rceil$.
2: Output: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_l$.
3: $\mathcal{S} \leftarrow \emptyset$, $R_0 \leftarrow \emptyset$.
4: for $r \leftarrow 0, \ldots, l$ do
5:     $R'_r \leftarrow R_r + \sum_{rk\delta+1 \leq j \leq (r+1)k\delta} D_j$.
6:     $\mathcal{S}_r \leftarrow OfflineAlgorithm\,(G, R'_r, \delta, k\delta)$.
7:     $R_{r+1} \leftarrow R'_r - min\left(R'_r, \sum_{(\alpha,M)\in\mathcal{S}_r} \alpha M\right)$, $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_r$.
8: end for
9: return $\mathcal{S}$
```

Proof of Theorem 3. 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 6. As we did in the proof of Theorem 6, 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 3.1 shows this setting. Using Lemma 1, there

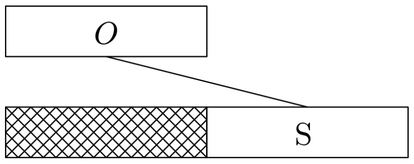

Figure 3.1: Basis of the induction. The crossed out block is the waiting period of our algorithm

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 3.2 for an illustration of this setting. Consider the new input sequence $\Gamma' =$

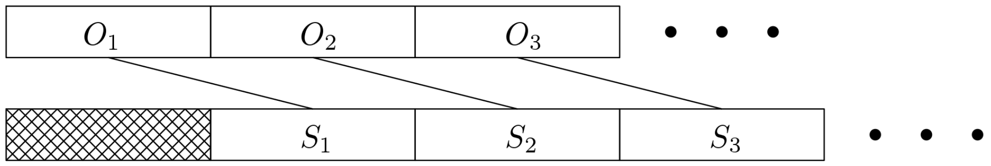

Figure 3.2: Step of the induction.

$\{R_1' \cup I_2, I_3, \ldots, I_t\}$. 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

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

Note that $S_i = S_{i-1}'$ for $i \geq 2$ and thus

$$f\left(\mathcal{S}'\right) = f\left(\mathcal{S} \setminus S_1\right) = f\left(\mathcal{S}\right) - f\left(S_1\right).$$

As in the proof of Lemma 6, 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

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

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

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

and

$$f\left(S_1\right) \geq \left(1 - \frac{2}{k}\right)\beta f\left(O_1\right).$$

Rewriting the first inequality we have

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

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

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

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

□

The running time of this algorithm is depended on the running time of the offline algorithm. Using the greedy algorithm the running time would be $O(d^2(\frac{n^5}{2})\frac{W}{\delta})$ for each step of the online algorithm. However, the better approximation will increase the time of the algorithm to include the Ellipsoid Method.

CHAPTER 4

CONCLUSION

In this thesis, we considered the problem of circuit switching. We considered this problem in two setting. The offline setting, where we know all the data in advance and have a limited amount of time to schedule as many configurations as we can and the online version in which the extra demand comes in at every step and accumulates to every data that we were not able to send before. We provided two algorithms for the Circuit Switch problems. Chapter 2 is a hybrid between greedy and a linear programming algorithm that will provide a $(1-{}^{1}\!/_{e}-\epsilon)$ approximation ratio (for any constant $\epsilon > 0$). This is an improvement over the previous result in this area. Whether or not the same approximation can be achieved faster is the subject of future work. We defined a novel online setting that is more practical in nature and provided a approximation algorithm with competitive ratio of $((e-1)/(2e-1)-\epsilon)$. Our algorithm is depended on the approximation algorithm of the offline setting. Therefore, any improvement on the algorithm in the offline setting will provide a better algorithm in the online version. The possibility of an algorithm that works only in the online setting regardless of the offline version was out of our scope and could be the topic of a future paper.

REFERENCES

[1] 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*, vol. 22, no. 2, pp. 498–511, 2014.

[2] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, “Helios: A hybrid electrical/optical switch architecture for modular data centers”, *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 339–350, 2010.

[3] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, “C-through: Part-time optics in data centers”, in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 40, 2010, pp. 327–338.

[4] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, “Firefly: A reconfigurable wireless data center fabric using free-space optics”, in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 44, 2014, pp. 319–330.

[5] S. Kandula, J. Padhye, and P. Bahl, “Flyways to de-congest data center networks”, 2009.

[6] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, “Mirror mirror on the ceiling: Flexible wireless links for data centers”, *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 443–454, 2012.

[7] C. Li, M. K. Mukerjee, D. G. Andersen, S. Seshan, M. Kaminsky, G. Porter, and A. C. Snoeren, “Using indirect routing to recover from network traffic scheduling estimation error”, in *Architectures for Networking and Communications Systems (ANCS), 2017 ACM/IEEE Symposium on*, IEEE, 2017, pp. 13–24.

[8] L. Liu, L. Gong, S. Yang, J. Xu, and L. Fortnow, “Better algorithms for hybrid circuit and packet switching in data centers”, *arXiv preprint arXiv:1712.06634*, 2017.

[9] S. B. Venkatakrishnan, M. Alizadeh, and P. Viswanath, “Costly circuits, submodular schedules and approximate carathéodory theorems”, *Queueing Systems*, pp. 1–37, 2018.

[10] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, “On service guarantees for input-buffered crossbar switches: A capacity decomposition approach by birkhoff and von neu-

mann”, in *1999 Seventh International Workshop on Quality of Service. IWQoS’99.(Cat. No. 98EX354)*, IEEE, 1999, pp. 79–86.

[11] S. Dughmi, “Submodular functions: Extensions, distributions, and algorithms. a survey”, *arXiv preprint arXiv:0912.0322*, 2009.

[12] M. Sviridenko, “A note on maximizing a submodular set function subject to a knapsack constraint”, *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004.

[13] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[14] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky, *et al.*, “Scheduling techniques for hybrid circuit/packet networks”, in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ACM, 2015, p. 41.

[15] X. Li and M. Hamdi, “On scheduling optical packet switches with reconfiguration delay”, *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1156–1164, 2003.

[16] S. 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*, ACM, 2015, pp. 361–369.

[17] F. Dufossé, K. Kaya, I. Panagiotas, and B. Uçar, “Further notes on birkhoff-von neumann decomposition of doubly stochastic matrices”, PhD thesis, Inria-Research Centre Grenoble–Rhône-Alpes, 2017.

[18] J. Kulkarni, E. Lee, and M. Singh, “Minimum birkhoff-von neumann decomposition”, in *International Conference on Integer Programming and Combinatorial Optimization*, Springer, 2017, pp. 343–354.

[19] V. Mirrokni, R. P. Leme, A. Vladu, and S. C.-w. Wong, “Tight bounds for approximate carath\’eodory and beyond”, *arXiv preprint arXiv:1512.08602*, 2015.

[20] T. Inukai, “An efficient ss/tdma time slot assignment algorithm”, *IEEE Transactions on Communications*, vol. 27, no. 10, pp. 1449–1455, 1979.

[21] B. Towles and W. J. Dally, “Guaranteed scheduling for switches with configuration overhead”, *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 835–847, 2003.

[22] A. Dasylva and R Srikant, “Optimal wdm schedules for optical star networks”, *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 446–456, 1999.

[23] S. Fu, B. Wu, X. Jiang, A. Pattavina, L. Zhang, and S. Xu, “Cost and delay tradeoff in three-stage switch architecture for data center networks”, in *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*, IEEE, 2013, pp. 56–61.

[24] S. Vargaftik, K. Barabash, Y. Ben-Itzhak, O. Biran, I. Keslassy, D. Lorenz, and A. Orda, “Composite-path switching”, in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, ACM, 2016, pp. 329–343.

[25] S. Khuller, A. Moss, and J. S. Naor, “The budgeted maximum coverage problem”, *Information processing letters*, vol. 70, no. 1, pp. 39–45, 1999.

[26] A. Badanidiyuru and J. Vondrák, “Fast algorithms for maximizing submodular functions”, in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2014, pp. 1497–1514.

[27] A. Ene and H. L. Nguyen, “A nearly-linear time algorithm for submodular maximization with a knapsack constraint”, *CoRR*, vol. abs/1709.09767, 2017. arXiv: 1709.09767.

[28] G Celik, S. C. Borst, P. A. Whiting, and E. Modiano, “Dynamic scheduling with reconfiguration delays”, *Queueing Systems*, vol. 83, no. 1-2, pp. 87–129, 2016.

[29] L. Georgiadis, M. J. Neely, L. Tassiulas, *et al.*, “Resource allocation and cross-layer control in wireless networks”, *Foundations and Trends® in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[30] R. Carr and S. Vempala, “Randomized metarounding”, in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 58–62.

[31] N. Andelman and Y. Mansour, “Auctions with budget constraints”, in *Scandinavian Workshop on Algorithm Theory*, Springer, 2004, pp. 26–38.