New Models and Algorithms for Throughput Maximization in Broadcast Scheduling

Chandra Chekuri [*] Avigdor Gal [†] Sungjin Im [‡] Samir Khuller [§] Jian Li [¶]
Richard McCutchen [||] Benjamin Moseley [**] Louiqa Raschid [††]

Abstract

In this paper we consider some basic scheduling questions motivated by query processing that involve accessing resources (such as sensors) to gather data. Clients issue requests for data from resources and the data may be volatile or changing which imposes temporal constraints on the delivery of the data. A proxy server has to compute a probing schedule for the resources since it can probe a limited number of resources at each time step. Due to overlapping client requests, multiple queries can be answered by probing the resource at a certain time. This leads to problems related to some well-studied broadcast scheduling problems. However, the specific requirements of the applications motivate some generalizations and variants of previously studied metrics for broadcast scheduling. We consider both online and offline versions of these problems and provide new algorithms and results.

[*]Department of Computer Science, University of Illinois, 201 N. Goodwin Ave., Urbana, IL 61801. chekuri@cs.illinois.edu. Partially supported by NSF grants CCF-0728782 and CNS-0721899.

[†]Faculty of Industrial Engineering & Management, Technion – Israel Institute of Technology, Technion City, Haifa 32000, Israel. avigal@ie.technion.ac.il

[‡]Department of Computer Science, University of Illinois, 201 N. Goodwin Ave., Urbana, IL 61801. im3@cs.illinois.edu. Supported mainly by a Samsung Fellowship, and partially by NSF grants CCF-0728782 and CNS-0721899.

[§]Department of Computer Science and UMIACS, University of Maryland College Park, MD 20742. samir@cs.umd.edu. Supported by NSF Awards CCF-0728839 and CCF-0937865, and a Google Research Award.

[¶]Department of Computer Science and UMIACS, University of Maryland College Park, MD 20742. lijian@cs.umd.edu

[||]Department of Computer Science, University of Maryland College Park, MD 20742. rmccutch@cs.umd.edu

[**]Department of Computer Science, University of Illinois, 201 N. Goodwin Ave., Urbana, IL 61801. bmosele2@cs.illinois.edu. Partially supported by NSF grant CNS-0721899.

[††]Robert H. Smith School of Business and UMIACS, University of Maryland College Park, MD 20742. louiqa@umiacs.umd.edu

1 Introduction

There is an explosion in the amount of data being produced, collected, disseminated and consumed. One important source of this data is from sensors that are being widely deployed to monitor and collect a variety of information, for example, weather and traffic. Another important source of data comes from individuals and entities publishing content on the web. Two aspects of the above type of data are the following. First, a consumer/client is typically interested only in some small subset of the available data that is relevant to her. Second, the data has temporal relevance and a client is also typically interested in data that is within some time interval of interest to her; for example traffic on a particular road during her commute window. These trends have necessitated a significant increase in the sophistication of data delivery capabilities to keep up with quantity of the data, and the need for client customization [37]. There is a large effort in several areas of computer science to address these issues. Typically, software called middleware, handles the interface between the clients and data sources. In this paper we are concerned with certain scheduling problems that arise in processing the queries of the clients.

Middleware primarily consists of proxy servers that collect client queries and access data sources (such as sensors) to answer queries [15, 22, 8, 34, 16, 38]. In this work we consider a basic and central question that arises when the queries are time sensitive (they also may be periodic) such as "Give me the reading of sensor A at 15 mins after the hour, every hour". The main challenge is to schedule probes to the data sources (e.g., sensors), to obtain the data at the desired time for the clients. Due to processing limitations, the proxy server is limited to probing only a small number of sensors at each time step (we assume for simplicity that it probes one sensor at each time step). However, by probing a sensor at a particular time, multiple overlapping queries requesting data from this sensor, can be answered.

More formally, there are clients that issue queries for data from a resource at a specific time, by specifying an interval of time when the resource should be queried. A central server collects all the queries and needs to design a schedule to probe the resources to answer client queries. When a resource is probed, several client queries can be answered. Typically, the queries are simple and so the computational requirements are minimal; hence we focus on the design of the probing schedule. For example, by identifying overlapping queries to the same resource we may be able to significantly reduce the number of times we query the sensors, since we can "piggyback" all the queries [36, 37, 38]. This overlapping nature of query processing, is very similar to the manner in which broadcast scheduling problems are approached [4, 28, 30, 24, 9]. However, the sensor probing application gives rise to new and interesting variants of broadcast scheduling problems that have been considered so far. In this work we focus on a collection of online and offline problems motivated by the above application.

In the broadcast scheduling literature, three objectives have been the focus of study: (i) minimizing average response time[1] [28, 2, 18, 19, 23, 9, 26] (and many others), (ii) minimizing maximum response time [4, 9, 11], and (iii) maximizing throughput [30, 7, 14, 41]. By response time of a request, we mean the time from the arrival of the request to when it is satisfied. The first two metrics apply to settings in which all requests are to be satisfied. The third metric is relevant in situations where requests may not be satisfied beyond a certain time; in particular, the following model has been studied. Each request has a release time and deadline and it can only be satisfied within its time window and the goal is to maximize the number/weight of satisfied requests. We next explain why these metrics are not directly suitable for our purposes.

In sensor probing, the requests are time sensitive which calls for a more nuanced view of "satisfying" a request. For example, if a client requests the temperature reading, or traffic conditions at 5:30pm, then we may satisfy this query by reporting the value at 5:33pm, this would have a latency of 3 minutes. Suppose we report the value at 5.40pm with a latency of 10 minutes; the data may still be useful to the client but perhaps less than reporting the value at 5.33pm. Finally, the data may be irrelevant if the latency is say more than 20 minutes. This example demonstrates the two aspects of interest in a schedule. We are interested in "completeness", the number of client requests that can be satisfied before their deadline. We are also interested in the "latency" of those requests that we do satisfy. As can be seen, previous metrics do not capture the combination of these metrics; minimizing average response time ignores deadlines and maximizing throughput with deadlines ignores the latency of satisfied requests .

In this paper we take two approaches to finding schedules that address both completeness and latency. In the first approach, we associate an arbitrary *time-dependent profit* function with each query. The profit function can take into account the impact of the latency on the value to the client. The goal then would be to find a schedule that maximizes

[1] Response time is also commonly referred to as flow time.

the total profit of the requests. This model captures the previously studied maximum throughput metric, but allows more control over the quality of the schedule for queries. We consider both offline and online settings and obtain several new results. In the second approach, we directly address the tradeoff between completeness and latency. In addition to satisfying as many requests as possible we hope to also satisfy them close to when the request arrived (the arrival time could be the ideal time when the sensor should be probed). We formalize this in the following way: Given a set of requests and a desired level of completeness, generate a schedule that minimizes latency of satisfied requests while achieving the required level of completeness.

Finally, we consider another variant of the maximum throughput problem that is relevant to our application domain. In some cases, it is perfectly reasonable to report the value "before" the arrival time of the request. In the same example above, the proxy server may have the value of a sensor measuring temperature that has been probed at 5.28pm while a request for the same sensor arrives at 5.30pm. The server can use the reading at 5.28pm to answer the query. We model this aspect in two ways: by relaxing the time window of interest to the client both forwards and backwards in time, and by considering unimodal profit functions.

All the problems we consider are NP-Hard in the offline setting via simple reductions to known results on broadcast scheduling [9]. We, therefore, focus on the design of an efficient approximation algorithms. We also consider online variants and use the standard competitive analysis framework; for some variants we analyse the algorithms in the resource augmentation framework [27] wherein the algorithm is given extra speed over the adversary. We give below a formal description of the problems considered in the paper, followed by our results.

1.1 Problem Definitions

For convenience we shall use the standard broadcast scheduling notation of referring to pages instead of referring to sensors. We are given a set of pages $\mathcal{P} = \{p_1, \dots, p_n\}$. We assume that time is slotted, $\mathcal{T} = \{1, 2, \dots, T\}$ where T is an upper bound on the schedule length. Suppose a client sends a request for page p, which arrives at time a and is associated with a deadline d. If the server broadcast p at some time slot t such that $a \leq t \leq d$, we say the request is *satisfied*. We assume the server can broadcast at most one page in a single time slot. We use $J_{p,i}$ to denote the ith request for page p, which has the arrival time $a_{p,i} \in \mathbb{Z}^+$ and the deadline $d_{p,i} \in \mathbb{Z}^+$. Sometimes, we will consider a generalized request which may be associated with more than one interval. As a unifying notation, we use $\mathcal{T}_{p,i}$ to denote the set of time slots associated with request $J_{p,i}$. For example, if $J_{p,i}$ has only one interval, then $\mathcal{T}_{p,i} = \{a_{p,i}, a_{p,i}+1, \dots, d_{p,i}\}$. In this paper, we study the following objective functions.

1. Maximizing throughput (MAX-THP): The objective is to maximize the total number of satisfied requests. In the weighted version of MAX-THP, each request $J_{p,i}$ has a weight $w_{p,i}$. In this case, the objective is to maximize the total weight of all satisfied requests.

2. Maximizing total profit (MAX-PFT): This is a significant generalization of MAX-THP. In a MAX-PFT instance, each request $J_{p,i}$ is associated with an arbitrary non-negative profit function $g_{p,i} : \mathcal{T} \to \mathbb{Z}^+$. The interpretation is that if the request $J_{p,i}$ obtains a value/profit of $g_{p,i}(t)$ if it is satisfied by the broadcast of p at time t. However, p may be broadcast multiple times during a schedule. In that case the request $J_{p,i}$ obtains a profit $\max_{t \in \mathcal{T}_p^A} g_{p,i}(t)$ where $\mathcal{T}_p^A$ is the set of time slots in which p was broadcast by a given schedule. The objective is to find a scheduling A such that the total profit is maximized.

3. Completeness-Latency tradeoff: We are given a MAX-THP instance and a completeness threshold $C \in (0, 1]$. The goal is to find a schedule that completes C fraction of the requests before their deadline and subject to that constraint, minimizes the latency of the completed requests.

1.2 Outline of Results

We obtain several results for the problems described above. We give a high-level description of these results below.

Maximizing Throughput and Profit: Recall that MAX-PFT is a significant generalization of MAX-THP. There is a $3/4$-approximation for the MAX-THP problem [24] via a natural LP relaxation. We adapt the ideas in [24] to obtain a $3/4$-approximation for the special case of MAX-PFT when the profit functions for each query are unimodal (see Section 3), which is of particular interest to our setting. Second, for the general MAX-PFT problem we obtain a $(1-1/e)$-approximation, again via the natural LP relaxation. In addition, we show that the MAX-PFT problem can be cast as a special case of submodular function maximization subject to a matroid constraint. This allows us to not only obtain a different $(1-1/e)$-approximation but also several generalizations and additional properties via results in [5, 13]. The connection also allows us to easily show that the greedy algorithm gives a $1/2$ approximation for MAX-PFT in the *online* setting, generalizing prior work that showed this for MAX-THP [30].

We also consider how the approximation ratios and competitive ratios for MAX-THP and MAX-PFT can be improved via resource augmentation and other relaxations. We show that there is a 2-speed 1-approximation for MAX-THP. Previously, such a result was known only if all requests could be scheduled in a fractional solution [9]. In the online-setting we show that the simple greedy algorithm with s-speed achieves a $s/(s+1)$ competitive ratio for MAX-PFT. In a different direction we consider relaxing the time window in MAX-THP and prove the following result. If there is a fractional schedule that satisfies all the client requests (obtained by solving the LP relaxation to the IP), then there is an integral schedule with the following property: each request $J_{p,i}$ is satisfied in a window $[a_{p,i}-L, d_{p,i}+L]$ where $L = d_{p,i}-a_{p,i}$ is the window length of $J_{p,i}$. In other words, by either left shifting the window or right shifting the window by its length, we can always satisfy the request.

Completion-Latency Tradeoff: We show that there is an interesting tradeoff that can be obtained between latency and completeness when each request has an associated deadline. Given a fractional LP solution (obtained by relaxing the IP) for minimizing the total latency subject to a certain completeness level, we show that we can use randomized rounding to obtain a schedule with the following properties: the expected completeness of the schedule is $\frac{3}{4}C$, where C is the completeness of the fractional schedule and the expected latency of the scheduled requests is $D(C)$ where $D(C)$ is the minimum fractional latency with completeness requirement C.

In addition to the above results we also prove an additional result of interest in broadcast scheduling. This concerns the problem of minimizing the maximum response time. The first-in-first-out (FIFO) algorithm is 2-competitive in the online setting [4, 9, 11] and this is also the best known off-line approximation known. Moreover, it is known that in the online setting no deterministic algorithm is $(2-\epsilon)$-competitive for any $\epsilon > 0$ [4, 9]. Here, we show that this same lower bound holds even for *randomized* online algorithms in the oblivious adversary model. The details of this result can be found in the Appendix A.

2 Preliminaries

Several of our results rely on the dependent randomized rounding framework of [24]. We first describe the LP relaxation for MAX-THP that is used as the basis for the rounding process.

2.1 An LP Relaxation for MAX-THP

We consider a natural integer programming (IP) formulation for MAX-THP. We use the indicator variable $Y_p^{(t)}$. $Y_p^{(t)} = 1$ if page p is broadcast in time-slot t and $Y_p^{(t)} = 0$ otherwise. In addition we define variables $X_{p,i}$ for the request $J_{p,i}$. This variable is 1 if and only if $J_{p,i}$ is satisfied.

$$\begin{aligned}
\text{maximize} \quad & \sum_{p,i} w_{p,i} X_{p,i} \qquad (1)\\
\text{subject to} \quad & \sum_{t \in \mathcal{T}_{p,i}} Y_p^{(t)} \geq X_{p,i} \ \forall p,t, \quad \text{If } p \text{ is not broadcast in } \mathcal{T}_{p,i},\ J_{p,i} \text{ cannot be satisfied,}\\
& \sum_{p} Y_p^{(t)} \leq 1,\ \forall t, \quad \text{One page broadcast at one time-slot ,}\\
& X_{p,i} \in \{0,1\}, \forall p,t, \ \ Y_p^{(t)} \in \{0,1\}, \forall p,t
\end{aligned}$$

By letting the domain of $X_{(p,i)}$ and $Y_p^{(t)}$ be $[0,1]$, we obtain the linear programming (LP) relaxation for the problem.

2.2 Dependent rounding scheme of [24]

We briefly describe the dependent randomized rounding method of [24]. Suppose we are given a bipartite graph (A, B, E) with bipartition (A, B). We are also given a value $x_{i,j} \in [0, 1]$ for each edge $(i, j) \in E$. The scheme in [24] provides a randomized polynomial-time algorithm that rounds each $x_{i,j}$ to a random variable $X_{i,j} \in \{0, 1\}$, in such a way that the following properties hold.

(P1): Marginal distribution. For every edge (i, j), $\Pr[X_{i,j} = 1] = x_{i,j}$.

(P2): Degree-preservation. Consider any vertex $i \in A \cup B$. Define its fractional degree d_i to be $\sum_{j:(i,j)\in E} x_{i,j}$, and integral degree D_i to be the random variable $\sum_{j:(i,j)\in E} X_{i,j}$. Then, $D_i \in \{\lfloor d_i \rfloor, \lceil d_i \rceil\}$. Note in particular that if d_i is an integer, then $D_i = d_i$ with probability 1.

(P3): Negative correlation. If $f = (i, j)$ is an edge, let X_f denote $X_{i,j}$. For any vertex i and any subset S of the edges incident on i:

$$\forall b \in \{0, 1\}, \ \Pr[\bigwedge_{f\in S} (X_f = b)] \le \prod_{f\in S} \Pr[X_f = b]. \tag{2}$$

In other words, the scheme takes as input a bipartite graph with values $0 \le x_{ij} \le 1$ associated with the edges (i, j) and rounds each edge to 0 or 1 (an edge is either dropped or chosen). The rounding method ensures that the probability of choosing edge (i, j) is exactly x_{ij}, and at the same time the (integral) degree of each node in the output is guaranteed to be $\lfloor d_i \rfloor$ or $\lceil d_i \rceil$. In addition, there is a *negative correlation property* (the property (P3)) which is important in some applications. We refer the reader to [24] for more details. In this paper we do not rely on (P3).

3 Throughput and Profit Maximization

This section is devoted to offline and online algorithms for MAX-THP and MAX-PFT.

3.1 Offline Algorithms

3.1.1 Maximizing the Total Profit

In this section, we consider the profit maximization (MAX-PFT) problem. Recall that in a MAX-PFT instance, each request $J_{p,i}$ is associated with a profit function $g_{p,i}(t) \ge 0$ that is an arbitrary non-negative function of the time it is satisfied. If a request page p is satisfied multiple times by a scheduling A, the profit we can get for p is the maximum one, i.e., $\max_{t\in\mathcal{T}_p^A} g_{p,i}(t)$. The objective is to find a scheduling A such that the total profit is maximized. Note that MAX-THP is just a special case of MAX-PFT where the profit function $g_{p,i}(t)$ is 1 for $a_{p,i} \le t \le d_{p,i}$.

First, we show how to reduce MAX-PFT to MAX-THP with weighted requests where each request may have multiple intervals. We use a simple slicing trick described as follows. Consider a single request $J_{p,i}$, and let $v_1 < v_2 < \ldots < v_r$ be the distinct nonnegative values taken on by its profit function $g_{p,i}$. Let $v_0 = 0$. We create r new requests for the throughput maximization instance, say $J_{p,i,j}, 1 \le j \le r$, which all require page p. $J_{p,i,j}$ has weight $v_j - v_{j-1}$ and intervals consisting of time slots $\{t \mid g_{p,i}(t) > v_{g-1}\}$. See Figure 1. It is not hard to show the following lemma.

Lemma 1 *The total (weighted) throughput of a schedule A for the constructed* MAX-THP *instance equals its total profit when interpreted as a schedule for the original* MAX-PFT *instance and vise versa.*

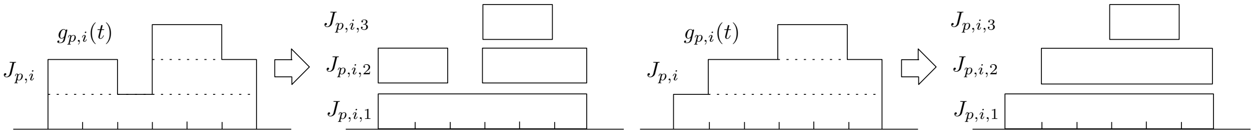

Figure 1: Illustrations of the slicing trick. The left hand side is a request with a general profit function and the right hand side is one with a unimodal profit function.

If each profit function is *unimodal*, meaning that it is non-decreasing up to a point and non-increasing after that point, We observe that the slicing trick should create requests each having only one request interval since the time slots $\{t \mid g_{p,i}(t) > v_{r-1}\}$ are consecutive (See the right hand side of Figure 1). Therefore, we can apply any approximation algorithm that works for the weighted throughput maximization problem with one interval for each request and obtain the same approximation ratio for profit maximization with unimodal profit functions. The best known approximation ratio for weighted throughput maximization is $3/4$ due to Gandhi et al. [24].

Theorem 1 *For arbitrary non-negative unimodal profit functions, there is a $3/4$-approximation for* MAX-PFT.

However, if the profit function is not unimodal, the resulting MAX-THP instance may contain requests that have multiple request intervals. Next, we show that a simple independent rounding scheme that gives a $1 - 1/e$-approximation for MAX-THP with each request associated with one or more intervals, which implies a $1 - 1/e$-approximation for MAX-PFT.

Let $x_{p,i}, y_p^{(t)}$ be the optimal fractional solution of LP 1. Consider the following simple independent rounding scheme: Consider each time slot t independently and choose exactly one page to broadcast. Page p is chosen with probability $y_p^{(t)}$. Note that this is feasible since $\sum_p y_p^{(t)} \leq 1$. We can easily lower bound the probability that a request is satisfied by the schedule produced by the independent rounding.

Lemma 2 *Using independent rounding, the probability that a request $J_{p,i}$ is satisfied is at least $(1 - 1/e)x_{p,i}$.*

Proof: For request $J_{p,i}$, we know that $x_{p,i} = \min(\sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)}, 1)$.

$$\Pr[J_{p,i} \text{ is satisfied}] = 1 - \Pr[J_{p,i} \text{ is not satisfied}] = 1 - \prod_{t \in \mathcal{T}_{p,i}} (1 - y_p^{(t)}) \quad \geq 1 - \prod_{t \in \mathcal{T}_{p,i}} e^{-y_p^{(t)}} = 1 - e^{-\sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)}}$$
$$\geq \min\Big(1 - \frac{1}{e}, (1 - \frac{1}{e}) \sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)}\Big) = (1 - \frac{1}{e}) x_{p,i}.$$

The first inequality follows from $1 - x \leq e^{-x}$ and the last is due to $1 - e^{-x} \geq (1 - 1/e)x \ \forall 0 \leq x \leq 1$. □

The expected total number of requests captured is thus

$$\sum_{p,i} \Pr[J_{p,i} \text{ is satisfied}] \geq (1 - \frac{1}{e}) \sum_{p,i} w_{p,i} x_{p,i} \geq (1 - \frac{1}{e}) OPT.$$

We thus conclude:

Theorem 2 *For any non-negative profit functions, there is a $1 - 1/e$-approximation for* MAX-PFT.

MAX-PFT **via Submodular set function maximization:** We described a $(1 - \frac{1}{e})$-approximation for MAX-PFT via an LP based approach. An alternative algorithm achieving the same ratio can also be obtained by casting MAX-PFT as a special case of the problem of maximizing a monotone submodular set function subject to a matroid constraint. For completeness we give a brief overview of matroid and submodular functions. For insufficient space, we only introduce the most relevant definitions and results. The interested reader is referred to [39] for basic definitions of submodular functions and [33, 21, 6, 40, 5] for old and new work on constrained submodular set function maximization.

First we give the definition of matroid. Let N be a finite set and $\mathcal{I}$ be a family of subsets of N. The pair $(N, \mathcal{I})$ is called matroid if $\mathcal{I}$ satisfies the following properties. (1) $\mathcal{I}$ is non-empty, (2) downward closed: if $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$, and (3) independent: if $A, B \in \mathcal{I}$ and $|A| < |B|$, then $A + x \in \mathcal{I}$ for some $x \in B \setminus A$. One special matroid is a partition matroid. In a *partition* matroid, N is partitioned into N_1, N_2, ... ,N_ℓ with associated integers $k_1, k_2, ..., k_\ell$, and $A \in \mathcal{I}$ if and only if $\forall i \ |A \cap N_i| \leq k_i$. Next we give the definition of monotone submodular set function $f : 2^N \rightarrow \mathcal{R}^+$. The function f is called submodular when $f(A + x) - f(A) \leq f(B + x) - f(B)$ for any $B \subseteq A$ and any $x \in N$. By monotonicity, we mean that if $B \subseteq A$ then $f(B) \leq f(A)$, and $f(\emptyset) = 0$. The problem of maximizing the submodular function f under the matroid constraint $(N, \mathcal{I})$ can be formulated as finding $A = \arg\max_{A' \in \mathcal{I}} f(A')$.

We interpret MAX-PFT as a special case of the above general problem in the following way. Let $N = \mathcal{P} \times \mathcal{T}$, where $\mathcal{P}$ is the set of pages and $\mathcal{T}$ is the set of times slots. Let $N_t = \mathcal{P} \times \{t\}$. Let $A \in \mathcal{I}$ iff $\forall t\ |A \cap N_t| \leq 1$. Notice that $(N, \mathcal{I})$ is a partition matroid. The function f is defined as follows: $f = \sum_{p,i} \max_{t:(p,t)\in A} g_{p,i}(t)$. It is not hard to see that f is a monotone submodular set function. It is known that maximizing a monotone submodular function can be approximated with factor $(1 - 1/e)$ under a matroid constraint [40, 5]. Using this reduction we can obtain a $(1 - \frac{1}{e})$-approximation for MAX-PFT.

The advantage of the alternative algorithm above is the following. Once the connection to submodular functions and matroid constraints is seen, one can readily obtain similar results for more general settings. For example, it is possible that a client request can be satisfied by different pages as long as they are similar. In this case, as long as, one is able to define an appropriate submodular profit function, one again obtains a $(1 - 1/e)$-approximation. Moreover, one can also impose additional constraints as long as they satisfy a matroid constraint; multiple matroid constraints can also be handled with some additional loss in the approximation. Finally, one can also obtain concentration bounds in some cases [13] and these can be useful in handling additional constraints. We defer a detailed description of some of these extensions to a later version of the paper.

3.1.2 A 2-Speed 1-Approximation for Throughput Maximization

In this section, our goal is to show a randomized 2-speed 1-approximation for throughput maximization. Here the objective is to satisfy as many requests as possibly by their deadline. Recall that in [9], it was shown that if there exists a fractional solution that satisfies all requests by their deadlines then there is a 2-speed algorithm that satisfies all requests by their deadline. To obtain a true 1-approximation, we need to also consider the case where the fractional solution does not have satisfy all request's by their deadline. Our analysis relies on the result of [9]. For completeness, we begin by showing that if there is a feasible fractional solution to LP (1) that satisfies all requests, there is a 2-speed integral scheduling that can also satisfy all requests. Then we show how to extend this to obtain the 2-speed 1-approximation by using dependent rounding.

Let $x_{p,i}, y_p^{(t)}$ be a fractional solution to the LP where all requests are satisfied by their deadline. We first construct a bipartite graph $G = (U, V, E)$ as follows. One partite set U consists of vertices that represent time slots. Let u_t denote the vertex in U corresponding to time t. For page p, we associate time slot t with an interval $I_{p,t} = [\sum_{t'=1}^{t-1} y_p^{(t')}, \sum_{t'=1}^{t} y_p^{(t')})$.

For each page p, the other partite set V consists of $\lceil 2\sum_t y_p^{(t)} \rceil$ vertices, each of which represents a set of consecutive time slots, called a window. We use $W_{p,i}$ and $v_{p,i}$ to denote the ith window for page p and the corresponding node V, respectively. We first associate $v_{p,i}$ with an interval $I(v_{p,i}) = [0.5 \times (i-1), 0.5 \times i)$. Window $W_{p,i}$ contains all time slots t such that $I_{p,t} \cap I(v_{p,i}) \neq \emptyset$. If $W_{p,i}$ contains time slot t, there is an edge $(u_t, v_{p,i})$. We repeat this for each page p. See Figure 2 for an illustration of the construction.

Now we augment the bipartite graph to get a network flow instance. We add a source $\mathbf{s}$, edges $(\mathbf{s}, u_t)$ with capacity 1 for $\forall t$, a sink $\mathbf{t}$ and edges $(v_{p,i}, \mathbf{t})$ with capacity $1/2$ for $\forall p, i$. First we can observe that there is $\mathbf{s}$-$\mathbf{t}$ flow f with flow value $\sum_{p,t} y_p^{(t)}$. In fact, just by letting $f(u_t, v_{p,i})$ be the measure (i.e., length) of $I_{p,t} \cap I(v_{p,i})$ and setting $f(\mathbf{s}, u_t) \forall t$ and $f(v_{p,i}, \mathbf{t}) \forall i$ accordingly, f is such a flow. For each page p, we delete the last window $W_{p, \lceil 2\sum_t y_p^{(t)} \rceil}$ if $f(v_{p, \lceil \sum_t y_p^{(t)} \rceil}, \mathbf{t}) < 1/2$. After this, we can see that f saturates all edges $(v_{p,i}, t) \forall p, i$.

Next, we double the capacities of all edges and find a maximum $\mathbf{s}$-$\mathbf{t}$ integral flow f'. This is possible since all capacities are now integral. The obtained integral flow can be interpreted as a 2-speed scheduling: If there is a unit of flow going from u_t to $v_{p,i}$, the server will broadcast p at time t. Since the capacity of $(\mathbf{s}, u_t)$ is 2, at most 2 pages are broadcast in one time slot. Note that all edges $(v_{p,i}, \mathbf{t})$ are saturated. This in turn means that for each window $W_{p,i}$, the server broadcasts p at least once in some time slot $t \in W_{p,i}$. For each request $J_{p,i}$, we know that $\sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)} \geq 1$. Therefore, some window $W_{p,j}$ is fully contained in $\mathcal{T}_{p,i} = \{a_{p,i}, \dots, d_{p,i}\}$ due to the construction of the windows. Hence, all requests are satisfied by the 2-speed schedule.

Now, we generalize the above idea to get a true 2-speed 1-approximation, that is a schedule such that the server broadcasts at most 2 pages in one time slot and satisfies at least the number of requests that can be satisfied by the

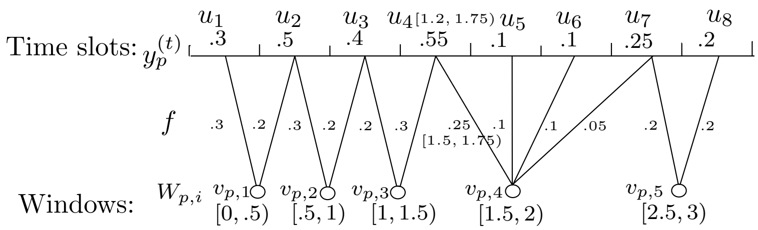

Figure 2: The construction of the bipartite graph $G(U, V, E)$. E.g. $I_{p,4} = [1.2, 1.75)$, $I(v_{p,4}) = [1.5, 2)$ and $I((u_4, v_{p,4})) = I_{p,4} \cap I(v_{p,4}) = [1.5, 1.75]$.

optimum 1-speed schedule. The idea is very simple, instead of scaling the capacities, we just take the bipartite graph G and the flow f, and do dependent rounding on G with values $2 \times f$. We notice that all f values defined on the edges of G are at most $1/2$. Therefore, $2 \times f$ are valid probabilities. Consider a request $J_{p,i}$ which is not completely satisfied, i.e., $\sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)} < 1$. In this case, $x_{p,i} = \sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)}$. It is easy to see that $\mathcal{T}_{p,i}$ is fully contained in two windows $W_{p,j}, W_{p,j+1}$ for some j. Let $y = \sum_{t \in \mathcal{T}_{p,i}} f(u_t, v_{p,j})$. By (**P2**) of the dependent rounding scheme, we know at most 1 edge incident on a window can be chosen. Therefore, for fixed p, j, the events that $(u_t, v_{p,j})$ is chosen are disjoint. Then, by (**P1**), we get that

$$\Pr[(u_t, v_{p,j}) \text{ is chosen for some } t \in \mathcal{T}_{p,i}] = 2 \sum_{t \in \mathcal{T}_{p,i}} y_p^{(t)} = 2y.$$

Similarly, we can show that $\Pr((u_t, v_{p,j+1}) \text{ is chosen for some } t \in \mathcal{T}_{p,i}) = 2(x-y)$. Therefore, $\Pr(J_{p,i} \text{ is satisfied}) \geq \max(2y, 2(x_{p,i}-y)) \geq x_{p,i}$. If $J_{p,i}$ is completely satisfied, we can use the previous argument, that is $\mathcal{T}_{p,i}$ fully contains some window $W_{p,j}$ and some edge incident to $v_{p,i}$ must be chosen. Again, we have $\Pr(J_{p,i} \text{ is satisfied}) = 1 = x_{p,i}$. Since we have shown that each request is satisfied with a probability no smaller than the probability that the request is satisfied in the fractional optimal solution, we obtain the following theorem.

Theorem 3 *There is a polynomial time* 2*-speed* 1*-approximation for* MAX-THP.

3.1.3 Throughput Maximization with a Relaxed Time Window

In this section, we assume that each request is fractionally fully satisfied by the optimal solution of LP(1), i.e., $x_{p,i} = 1 \ \forall p, i$. Suppose a request $J_{p,i}$ arrives at time $a_{p,i}$ with deadline $d_{p,i}$ (associated with the window $[a_{p,i}, d_{p,i}]$), then we construct an integral schedule such that this request is satisfied within the window $[a_{p,i} - l_{p,i}, d_{p,i} + l_{p,i}]$ where $l_{p,i} = d_{p,i} - a_{p,i} + 1$ is the length of the window $\mathcal{T}_{p,i}$. By left shifting the window or right shifting the window by its length, we can satisfy the request. A shifting, or expanding of the window is necessary and we refer to this as a relaxed schedule since it satisfies all the requests in a relaxed manner, and the client request is satisfied at a time approximately close to the desired window of time. For a given instance, we consider a fractional solution which, by assumption, satisfies all requests before their deadlines.

Starting from the instance I that has a fractional solution in which every request is satisfied, we will create an instance $\mathcal{I}$ which is a subset of the requests such that finding an integral solution for $\mathcal{I}$ will also immediately lend a *relaxed integral solution* to the instance $\mathcal{I}$. We focus on a single page p. Order all the client requests for page p in order of non-decreasing window length. Initially $\mathcal{I}$ is the empty set of requests. We try to insert each request (in non-decreasing window length order) into set $\mathcal{I}$, and as long as it does not overlap with a request already inserted into $\mathcal{I}$, we insert it. This will give us a collection of non-overlapping requests for page p. We do this filtering for every possible page. This gives us a fractional solution in which all requests for the same page are non-overlapping and completely satisfied. Using flow based methods[2] it is easy to convert this to an integral solution that satisfies all the requests. Client requests in $\mathcal{I}$ are clearly satisfied (integrally) within their intervals. Each client request $J_{p,i}$ that was

[2]This involves the same technique as used for converting a fractional matching in a bipartite graph to an integral matching [23].

not chosen in $\mathcal{I}$ overlapped with a chosen request with a smaller window size. Thus it is also satisfied in the integral solution within time $[a_{p,i} - l_{p,i}, d_{p,i} + l_{p,i}]$,i.e., satisfied within the relaxed deadlines. We thus conclude:

Theorem 4 *Suppose there is a fractional schedule that satisfies all requests. We can convert the fractional solution to an integral one in polynomial time such that each request $J_{p,i}$ can be satisfied in the relaxed window $[a_{p,i} - l_{p,i}, d_{p,i} + l_{p,i}]$ where $l_{p,i} = d_{p,i} - a_{p,i} + 1$ is the length of the window $\mathcal{T}_{p,i}$.*

3.2 Online Algorithms

In this section we revisit the problem MAX-PFT, but now in the online setting. In the *online* setting, a request is not known to the server until it arrives. As previously discussed in Section 3, maximizing the total profit can be interpreted as maximizing a monotone submodular function subject on a matroid. It is known that a simple greedy algorithm gives 2-approximation for such a problem [33]. Further, the greedy algorithm can be interpreted as an online algorithm in this setting. Thus we can easily obtain a 2-competitive algorithm for MAX-PFT. For the more restricted setting MAX-THP, [30] gave a 2-approximation. Here we show that the greedy algorithm's performance improves in the resource augmentation model when the algorithm is given a speed larger than 1. There is no natural way to interpret resource augmentation in the general framework of submodular set function maximization subject to a matroid constraint. We therefore resort to a direct analysis.

We will be considering a resource augmentation analysis [27]. In this analysis, the online algorithm is given s-speed and compared to a 1 speed optimal solution. For some objective function, we say that an algorithm is s-speed c-competitive if the algorithm's objective is within a c factor of the optimal solution's objective. Due to general form of the profit functions, we need more formal definitions and notation. Recall that each request is associated with its profit function $g_{p,i}(t)$. For simplicity, we assume $g_{p,i}(t) = 0$ for any time $t < a_{p,i}$. We define $m_{p,i}(t)$, the *so-far-gained* profit at time t for $J_{p,i}$, to be the maximum profit $J_{p,i}$ witnessed at any time when page p is broadcasted before time t, i.e. $m_{p,i}(t) = \max_{t':t' \in \mathcal{T}_p, t' \leq t} g_{p,i}(t')$. For request $J_{p,i}$ let the *final* profit $m_{p,i}$ be $\max_t m_{p,i}(t)$. Our objective is to maximize the total final profits. Let $\alpha_{p,i}(t)$, the *additional* profit at time t for $J_{p,i}$ be $\max(g_{p,i}(t) - m_{p,i}(t-1), 0)$. Note that if page p is broadcasted at time t, then $\alpha_{p,i}(t) = m_{p,i}(t) - m_{p,i}(t-1)$. To denote the values corresponding to OPT, the superscript * will be used. For example, $m^*_{p,i}$ is the final profit for $J_{p,i}$ by OPT.

We describe our greedy algorithm *Maximum Additional Profits First* (for short, MAPF) which is given an integer speed $s \geq 1$. As implied in its name, the algorithm MAPF broadcasts s pages which give the maximum additional profits by broadcasting.

Algorithm: MAPF
• At any time t, broadcast s pages which give the maximum additional profits.

We prove the following theorem.

Theorem 5 MAPF *is s-speed $(1 + 1/s)$-competitive online algorithm for* MAX-PFT.

Corollary 1 *There is a s-speed $(1 + 1/s)$-competitive algorithm for* MAX-THP.

For ease of notation, we let OPT and MAPF denote also the total (final) profits by the optimal solution and the algorithm MAPF, respectively. Formally, OPT $= \sum_{p,i} m^*_{p,i}$ and MAPF $= \sum_{p,i} m_{p,i}$. For each request $J_{p,i}$, consider the increase of the so-far-gained profit by OPT over the final profit by MAPF, formally $\max(m^*_{p,i}(t) - \max(m^*_{p,i}(t-1), m_{p,i}), 0)$. Let $\text{OPT}'(t)$ denote the sum of the increases at time t, i.e. $\text{OPT}'(t) = \sum_{p,i} \max(m^*_{p,i}(t) - \max(m^*_{p,i}(t-1), m_{p,i}), 0)$. We let $\text{OPT}(t)$ and $\text{MAPF}(t)$ denote the increase in the so-far-gained profits achieved by OPT and MAPF at time t, respectively. That is, $\text{OPT}(t) = \sum_{p,i} m^*_{p,i}(t) - m^*_{p,i}(t-1)$ and $\text{MAPF}(t) = \sum_{p,i} m_{p,i}(t) - m_{p,i}(t-1)$. The analysis is based on the following inequalities.

$$\begin{aligned} \text{OPT} &\leq \text{MAPF} + \textstyle\sum_t \text{OPT}'(t) && \text{[Lemma 3]} \\ &\leq \text{MAPF} + \textstyle\sum_t (1/s)\text{MAPF}(t) && \text{[Lemma 4]} \\ &= (1 + 1/s)\text{MAPF} \end{aligned}$$

The first inequality holds because all extra profits by OPT over MAPF are counted in $\sum_t \mathrm{OPT}'(t)$. The second inequality is due to the property of MAPF. If $\mathrm{OPT}'(t)$, the additional profit OPT gains over the final profit by MAPF, is non-zero, it implies that MAPF did not broadcast the same page q that OPT did at time t. Thus it follows that MAPF broadcast s other pages which give at least as good additional profit as page q. Then we have that $s\mathrm{OPT}'(t) \leq \mathsf{MAPF}(t)$. The formal proofs can be found in the following lemmas.

Lemma 3 $\mathrm{OPT} \leq \mathsf{MAPF} + \sum_t \mathrm{OPT}'(t)$.

Proof: By the definition of OPT and simple algebra,

$$\begin{aligned}
\mathrm{OPT} &= \sum_{p,i} m^*_{p,i} \leq \sum_{p,i}[m_{p,i} + \max(m^*_{p,i} - m_{p,i}, 0)] = \mathsf{MAPF} + \sum_{p,i} \max(m^*_{p,i} - m_{p,i}, 0) \\
&\leq \mathsf{MAPF} + \sum_{p,i} \max(\sum_t [m^*_{p,i}(t) - m^*_{p,i}(t-1)] - m_{p,i}, 0).
\end{aligned}$$

By the definition of $\mathrm{OPT}'(t)$, we have that $\sum_t \mathrm{OPT}'(t) = \sum_t \sum_{p,i} \max(m^*_{p,i}(t) - \max(m^*_{p,i}(t-1), m_{p,i}), 0)$. By changing the order of summation, it is easy to see that we only need to show

$$\max(\sum_t [m^*_{p,i}(t) - m^*_{p,i}(t-1)] - m_{p,i}, 0) \leq \sum_t \max(m^*_{p,i}(t) - \max(m^*_{p,i}(t-1), m_{p,i}), 0).$$

This can be easily shown using the fact that $m^*_{p,i}(t) \geq 0$ is non-decreasing. $\square$

Lemma 4 *For any integer speed $s \geq 1$, $s\mathrm{OPT}'(t) \leq \mathsf{MAPF}(t)$.*

Proof: We assume that $\mathrm{OPT}'(t) \neq 0$, since otherwise the inequality is trivial. Let q be the page OPT broadcasts at time t. By $m^*_{q,i}(t) = \max(g_{q,i}(t), m^*_{q,i}(t-1))$, we have that

$$\mathrm{OPT}'(t) = \sum_i \max(\max(g_{q,i}(t), m^*_{q,i}(t-1)) - \max(m^*_{q,i}(t-1), m_{q,i}), 0). \tag{3}$$

Without loss of generality, we assume that MAPF broadcasts s distinct pages $\mathcal{P}(t)$.

Notice that

$$\mathsf{MAPF}(t) = \sum_{p\in\mathcal{P}(t)} \sum_i m_{p,i}(t) - m_{p,i}(t-1) = \sum_{p\in\mathcal{P}(t)} \sum_i \alpha_{p,i}(t). \tag{4}$$

Note that MAPF did not broadcast the page q which OPT did at time t, i.e. $q \notin \mathcal{P}(t)$; otherwise each term in the sum in (3) is 0, since $m_{q,i} \geq g_{q,i}(t)$. We now show that

$$\mathrm{OPT}'(t) \leq \sum_i \alpha_{q,i}(t). \tag{5}$$

Note that the right hand side is the additional profits which MAPF could have achieved by broadcasting page q at time t. By simple algebra and (3), (5) is reduced to

$$\max(\max(g_{q,i}(t), m^*_{q,i}(t-1)) - \max(m^*_{q,i}(t-1), m_{q,i}), 0) \leq \max(g_{q,i}(t) - m_{q,i}(t-1), 0).$$

This can be easily verified using the fact that $m_{q,i} \geq m_{q,i}(t-1)$.

Since MAPF broadcasts the s pages in $\mathcal{P}(t)$ different from q, by the property of MAPF,

$$\forall p \in \mathcal{P}(t), \sum_i \alpha_{q,i}(t) \leq \sum_i \alpha_{p,i}(t). \tag{6}$$

Combining (4), (5) and (6) completes the proof.

We also prove that the analysis of MAPF is tight in the restricted setting $\mathsf{MAX\text{-}THP}$. Note that in the setting MAPF transmits s pages which satisfy the largest number of unsatisfied requests at each time. Due to space constraints, we defer the proof to Appendix B.

Theorem 6 *For any $\epsilon > 0$ and speed $s \geq 1$, MAPF is not s-speed $(1 + 1/s - \epsilon)$-competitive for $\mathsf{MAX\text{-}THP}$.*

For $s = 1$, for any $\epsilon > 0$, there is a lower bound of $(2 - \epsilon)$ on the competitive ratio of any online algorithm, even if it is randomized [30].

4 Minimizing Latency subject to Completeness Requirements

In this section, we consider the problem of minimizing the total latency of the completed requests subject to completeness requirements. Recall that the input of this problem is a MAX-THP instance and a completeness threshold $C \in (0, 1]$, and the objective is to find a schedule that completes C fraction of the requests and minimizes the latency of the completed requests.

We set up the integer linear program as follows. We use some indicator variables: $X_{p,i}^{(t)} = 1$ indicates job $J_{p,i}$ is captured at time t. $Y_p^{(t)} = 1$ indicates the server broadcast page p at time t. Let N denote the total number of requests.

$$\begin{aligned}
\text{minimize} \quad & \sum_{p,i} \sum_{t=a_{p,i}}^{d_{p,i}} X_{p,i}^{(t)} \cdot (t - a_{p,i}) && (7)\\
\text{subject to} \quad & \sum_{t=a_{p,i}}^{d_{p,i}} X_{p,i}^{(t)} \leq 1 \ \text{ Each request satisfied at most once} \\
& \sum_{p} Y_p^{(t)} \leq 1 \ \text{ One page broadcast at time-slot } t \\
& X_{p,i}^{(t)} \leq Y_p^{(t)} \ \text{ If page is not broadcast, then the request cannot be satisfied} \\
& \sum_{p,i} \sum_{t=a_{p,i}}^{d_{p,i}} X_{p,i}^{(t)} \geq CN \ \text{ Completeness requirements} \\
& X_{p,i}^{(t)} \in \{0,1\}, \ \ Y_p^{(t)} \in \{0,1\}
\end{aligned}$$

We replace the integer constraints $X_{p,i}^{(t)} \in \{0, 1\}, \ \ Y_p^{(t)} \in \{0, 1\}$ by $X_{p,i}^{(t)} \in [0, 1], \ \ Y_p^{(t)} \in [0, 1]$ to obtain the LP relaxation. Suppose $x_{p,i}^{(t)}, y_p^{(t)}$ is the optimal fractional solution for this LP. The main result in this section is the following theorem which reveals an interesting tradeoff that can be obtained between latency and completeness.

Theorem 7 *Given a fractional LP solution, there is a polynomial time randomized algorithm that can obtain a schedule such that the expected completeness of the schedule is $\frac{3}{4}C$ and the expected latency of the scheduled requests is at most $D(C)$ where $D(C)$ is the minimum fractional latency with completeness requirement C.*

Before giving the algorithm that achieves the bound claimed above, we first analyze a simple independent rounding scheme with slightly worse bound. The first step of the scheme is similar to the one we used for MAX-THP in Section 3. The first step completely decides which requests are satisfied or not. Recall that we are making a tradeoff between the total latency and the completeness. Thus we will exclude some requests, even though they are satisfied, which could incur high latency. To this end, we will say we *capture* a satisfied request, if we include it to our solution (thus counting toward the desired completeness).

Algorithm: LATENCY/COMPLETENESS-IND-ROUND

1. Consider each time slot t independently. Page p is broadcast at t with probability $y_p^{(t)}$.
2. Consider each request $J_{p,i}$ independently. If p is broadcast in at least one time-slot in the request window, let t be the earliest such slot. With probability $x_{p,i}^{(t)}/y_p^{(t)}$, capture the request, otherwise leave it uncaptured.

Note that the probability $x_{p,i}^{(t)}/y_p^{(t)}$ is well defined; $x_{p,i}^{(t)} \leq y_p^{(t)}$, and $y_p^{(t)} > 0$ since p was broadcasted at time t. Let $x_{p,i} := \sum_t x_{p,i}^{(t)}$. Let $T_{p,i}$ be the first time at which $\sum_{t=a_{p,i}}^{T_{p,i}} y_p^{(t)} \geq x_{p,i}$. Observe that given the $y_p^{(t)}$ and $x_{p,i}$ values, the optimum (minimum latency) choice of the $x_{p,i}^{(t)}$ values will set $x_{p,i}^{(t)} = y_p^{(t)}$ for $t < T_{p,i}$, $x_{p,i}^{(t)} = 0$ for $t > T_{p,i}$, and $0 < x_{p,i}^{(T_{p,i})} < y_p^{(T_{p,i})}$. Thus, the probability in the second step is 1 for $t < T_{p,i}$ and 0 for $t > T_{p,i}$. The purpose

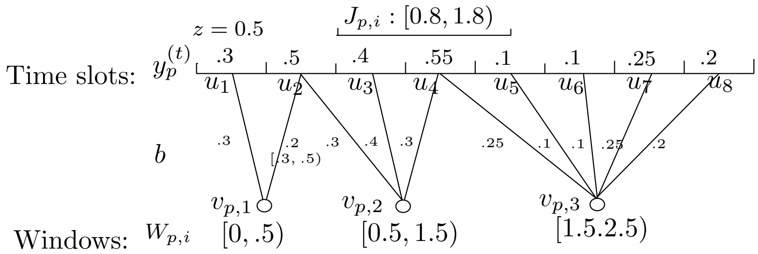

Figure 3: The construction of the bipartite graph $G(U, V, E)$. We choose $z = 0.5$. E.g. $I_{p,2} = [0.3, 0.8)$, $I(v_{p,1}) = [0, 0.5)$ and $I((u_2, v_{p,1})) = I_{p,2} \cap I(v_{p,1}) = [0.3, 0.5]$. Assume that $\mathcal{T}_{p,i} = \{3, 4, 5, 6\}$ for job $J_{p,i}$. Therefore, $x_{p,i} = 1$, $I(J_{p,i}) = [0.8, 1.8)$. and $E_{p,i} = \{(u_3, v_{p,2}), (u_4, v_{p,2}), (u_4, v_{p,3}), (u_5, v_{p,3})\}$.

of this step is to drop a request in the integral solution if it is satisfied within its window but the broadcast comes so late that it would contribute "too much" to the latency of the solution. Denote by OPT the optimal latency under the completeness constraint, that is at least C fraction of requests are satisfied. The performance of the independent rounding scheme is given by the following lemma. The proof can be found in Appendix B.

Lemma 5 *Given a fractional LP solution, the independent rounding scheme can obtain a schedule such that the expected completeness of the schedule is $(1 - \frac{1}{e})C$ and the expected latency of the scheduled requests is at most $D(C)$ where $D(C)$ is the minimum fractional latency with completeness requirement C.*

Now we improve the expected completeness to $\frac{3}{4}C$ using the dependent rounding technique. The bipartite graph construction in the first step is the same as the one used for throughput scheduling in [24], however we apply it to a slightly different LP formulation and also analyze the latency.

Algorithm: LATENCY/COMPLETENESS-DEP-ROUND

1. Construct a bipartite graph $G = (U, V, E)$ as follows. $U = \{u_t\}_t$ where u_t is a vertex corresponding to time slot t. For each page p group time slots into some number of windows, $W_{p,j}, 1 \leq j \leq m_p$ as follows. Choose $z \in (0, 1]$ uniformly at random. Each window is a set of time slots. For each window $W_{p,i}$, there is a vertex $v_{p,i} \in V$. Associate $v_{p,i}$ with an interval $I(v_{p,i})$ as follows: $I(v_{p,1}) = [0, z)$ and $I(v_{p,i}) = [i - 2 + z, i - 1 + z)$ for $i \geq 2$. For page p, also associate each time slot t with an interval $I_{p,t} = [\sum_{t'=1}^{t-1} y_p^{(t')}, \sum_{t'=1}^{t} y_p^{(t')})$. Window $W_{p,i}$ contains all time slots t such that $I_{p,t} \cap I(v_{p,i}) \neq \emptyset$. If $t \in W_{p,i}$, there is an edge $(u_t, v_{p,i})$. Associate to edge $e = (u_t, v_{p,i})$ an interval $I(e) = I_{p,t} \cap I(v_{p,i})$ and let $b(u_t, v_{p,i}) = |I(e)|$ where $|I(e)|$ is the length of $I(e)$. Repeat the above construction for each page p.

2. Perform dependent rounding in G with b as the probabilities defined on the edges. If an edge $(v_{p,i}, u_t)$ gets chosen in the rounded solution, then schedule page p at time t.

3. Consider each request $J_{p,i}$ independently. Associate to $J_{p,i}$ an interval $I(J_{p,i}) = [\sum_{t=1}^{a_{p,i}} y_p^{(t)}, \sum_{t=1}^{a_{p,i}} y_p^{(t)} + x_{p,i})$. Let $E_{p,i}$ be the set of edges e such that $I(e) \cap I(J_{p,i}) \neq \emptyset$. If any edge in $E_{p,i}$ is rounded up to 1 in the previous step, let e be the earliest such slot. With probability $\frac{|I(J_{p,i}) \cap I(e)|}{|I(e)|}$, capture the request using this broadcast, otherwise leave it uncaptured.

See Figure 3 for an example of the construction. In the example, if $(u_5, v_{p,3})$ is rounded up to 1 and $(u_3, v_{p,2})$ and $(u_4, v_{p,2})$ are rounded to 0 in step 2, we capture $J_{p,i}$ with probability $0.05/0.1 = 1/2$ in step 3. We now show the performance of the algorithm.

Proof of Theorem 7: Consider a particular request $J_{p,i}$. We first show that the probability that $J_{p,i}$ is scheduled is at least $\frac{3}{4}x_{p,i}$. We know that $x_{p,i} = \sum_{t \in \mathcal{T}_{p,i}} x_{p,i}^{(t)} \leq 1$. Thus, for any $z \in (0, 1]$, $I(J_{p,i})$ is fully contained in two intervals $I(v_{p,j}), I(v_{p,j+1})$ for some j. We claim that, conditioning on the choice z, $J_{p,i}$ is satisfied with probability at least

$$\max(|I(J_{p,i}) \cap I(v_{p,j})|, |I(J_{p,i}) \cap I(v_{p,j+1})|)$$

Since there is at most 1 edge adjacent to $v_{p,i}$ rounded to 1, the events that $J_{p,i}$ is satisfied by any of those edges are disjoint. Therefore, the probability $J_{p,i}$ is satisfied by any of those is

$$\sum_{e\in\partial(v_{p,i})}\Pr[e \text{ is rounded to } 1 \text{ and } J_{p,i} \text{ is captured by } e] = \sum_{e\in\partial(v_{p,i})}|I(e)|\frac{|I(J_{p,i})\cap I(e)|}{|I(e)|}$$
$$= \sum_{e\in\partial(v_{p,i})}|I(J_{p,i})\cap I(e)| = |I(J_{p,i})\cap \bigcup_{e\in\partial(v_{p,i})} I(e)| = |I(J_{p,i})\cap I(v_{p,j})|.$$

where $\partial(v)$ denote the set of edges incident on v. Hence, the claim holds.

Since the initial z was chosen uniformly randomly, the probability that $I(J_{p,i})$ is fully contained in one interval $I_{p,j}$ is $1-x_{p,i}$. In this case, the probability $J_{p,i}$ is satisfied is $|I(J_{p,i})\cap I(v_{p,j})| = x_{p,i}$. If $I(J_{p,i})$ overlaps with two intervals, $J_{p,i}$ is satisfied with probability at least $\max(\alpha, x_{p,i}-\alpha)$ conditioning on $|I(J_{p,i})\cap I(v_{p,j})| = \alpha$. Therefore, we have that

$$\Pr[J_{p,i} \text{ is satisfied}] = \int_0^{x_{p,i}} \max(\alpha, x_{p,i}-\alpha)d\alpha + (1-x_{p,i})x_{p,i} = \frac{3}{4}x_{p,i}^2 + (1-x_{p,i})x_{p,i} = x_{p,i} - \frac{1}{4}x_{p,i}^2 \geq \frac{3}{4}x_{p,i}.$$

To prove the second part, we only need to show that $\Pr(D(J_{p,i}) = t - a_{p,i}) \leq x_{p,i}^{(t)}$ which can be seen from that

$$\Pr(D(J_{p,i}) = t-a_{p,i}) = \Pr[\bigcup_j (J_{p,i} \text{ is captured by } (u_t, v_{p,j})] \leq \sum_j \Pr[J_{p,i} \text{ is captured by } (u_t, v_{p,j})]$$
$$\leq \sum_j |I((u_t, v_{p,j}))|\frac{|I(J_{p,i})\cap I((u_t,v_{p,j}))|}{|I((u_t,v_{p,j}))|} = |I(J_{p,j})\cap I_{p,t}| = x_{p,i}^{(t)}$$

The last equality follows from the fact that $x_{p,i}^{(t)} = y_p^{(t)}$ for $t < T_{p,i}$. □

5 Conclusions

In this paper we assumed that each query required access to a single resource. However, often answering client queries can be complex and may involve access to multiple sources and/or dependency among them. For instance, a query may read "give me the temperature at point A within next hour, and the temperature at point B within 10 minutes after sending the reading of A". Incorporating such queries seems to be an interesting and rich future direction.

Note that our dependent rounding algorithm in Section 4 only provides *expected* guarantees for both completeness and tradeoff. Unlike the case where there is only one objective, such a result does not necessarily imply the existence of a deterministic solution that achieves the claimed bounds for both objectives. We leave the problem of derandomizing our algorithm as an interesting open problem.

In addition, we are exploring an understanding of the limitations of LP rounding for the completeness-latency tradeoff question. For example, using an integrality gap γ (currently $\gamma = \frac{12}{13}$) for the LP [9] for scheduling with windows, we can show that to get better completeness, we will have to pay significantly on the latency using LP-rounding methods.

References

[1] D. Aksoy, and M. Franklin. RxW: A scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Transactions On Networking*, Volume 7, Number 6:846-860, 1999.

[2] N. Bansal, D. Coppersmith, and M. Sviridenko. Improved approximation algorithms for broadcast scheduling. *Proceedings of 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 344–353, 2006.

[3] N. Bansal, R. Krishnaswamy and V. Nagarajan. Better Scalable Algorithms for Broadcast Scheduling. *ICALP*, 2010.

[4] Y. Bartal and S. Muthukrishnan. Minimizing maximum response time inscheduling broadcasts. *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 558–559, 2000.

[5] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak. Maximizing a monotone submodular set function subject to a matroid constraint *SIAM J. on Computing*, to appear.

[6] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak. Maximizing a submodular set function subject to a matroid constraint *IPCO*, pp. 182–196, 2007.

[7] W. Chan, T. Lam, H. Ting, and P. Wong. New Results on On-Demand Broadcasting with Deadline via Job Scheduling with Cancellation. *COCOON*, pp. 210–218, 2004.

[8] D. Carney, S. Lee and S. Zdonik. Scalable Application-Aware Data Freshening. *ICDE*, pp. 481–492 (2003).

[9] J. Chang, T. Erlebach, R. Gailis and S. Khuller. Broadcast Scheduling: Algorithms and Complexity. *Proc. of 19th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 473–482, 2008.

[10] M. Charikar, S. Khuller. A robust maximum completion time measure for scheduling. *Proceedings of 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 324–333, 2006.

[11] C. Chekuri, S. Im and B. Moseley. Minimizing Maximum Response Timeand Delay Factor in Broadcast Scheduling. *ESA*, pp. 444–455, 2009.

[12] C. Chekuri, S. Im and B. Moseley. Longest Wait First for Broadcast Scheduling. *WAOA*, 2009.

[13] C. Chekuri, J. Vondrak and R. Zenklusen. Dependent Randomized Rounding for Matroid Polytopes and Applications. Manuscript, 2009. arXiv:0909.4348.

[14] M. Chrobak, C. Dürr, W. Jawor, L. Kowalik and Maciej Kurowski. A Note on Scheduling Equal-Length Jobs to Maximize Throughput. *J. of Scheduling*, Vol. 9(1):71–73, (2006).

[15] P. Deolasee, A. Katkar, P. Panchbudhe, K. Ramamritham and P. Shenoy. Adaptive Push-Pull: Deisseminating Dynamic Web Data. *Proc. of WWW*, (2001).

[16] J. Eckstein, A. Gal and S. Reiner. Monitoring an Information Source under a Politeness Constraint. *INFORMS Journal on Computing*, Vol. 20(1):3–20, (2007).

[17] J. Edmonds. Scheduling in the dark. *Theor. Comput. Sci.*, Vol. 235(1):109–1183, (2000).

[18] J. Edmonds and K. Pruhs. Multicast pull scheduling: when fairness is fine. *Proc. of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 421–430, 2002.

[19] J. Edmonds and K. Pruhs. A maiden analysis of longest wait first. *Proc. of 15th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 811–820, 2004.

[20] T. Erlebach, A. Hall. NP-Hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *Proc. of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 194–202, 2002.

[21] M. L. Fisher, G. L. Nemhauser and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions - II. *Math. Prog. Study*, 8:73–87, 1978.

[22] A. Gal and J. Eckstein. Managing Periodically Updated Data in Relational Databases: A Stochastic Modeling Approach. *Journal of the ACM*, Vol. 48(6):1141–1183, (2001).

[23] R. Gandhi, S. Khuller, Y. Kim, and Y.C. Wan. Algorithms for minimizing response time in broadcast scheduling. In *Proc. Ninth Conference on Integer Programming and Combinatorial Optimization* (May 2002), vol. 2337 of *Lecture Notes in Computer Science*, Springer, pp. 415–424.

[24] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM*, Vol 53(3):324–360, (2006). Prelminary version in *Proc. of IEEE FOCS*, 2002.

[25] P. Goundan and A. Schulz. Revisiting the greedy approach to submodular set function maximization *Manuscript*, 2007.

[26] S. Im and B. Moseley. An Online Scalable Algorithm for Average Flow Time in Broadcast Scheduling. *SODA*, 2010.

[27] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, Vol 47(4):617-643, (2000).

[28] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. *European Symposium of Algorithms*, LNCS 1879, Springer-Verlag, pp. 290–301, 2000.

[29] A. Kamath, R. Motwani, K. Palem, and P. Spirakis Tail Bounds for Occupancy and the Satisfiability Threshold Conjecture. Random Structures and Algorithms 7 (1995). 59-80.

[30] J. Kim and K. Chwa. Scheduling broadcasts with deadlines. *Theor. Comput. Sci.*, Vol 325:479–488, (2004).

[31] R. Motwani and P. Raghavan. Randomized Algorithms. *ACM Comput. Surveys*, Vol. 28(1):33–37 (1996).

[32] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[33] G. L. Nemhauser, L. A. Wolsey and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, Vol 14(1):265–294, (1978).

[34] S. Pandey, K. Dhamdhere and C. Olston. WIC: A General-Purpose Algorithm for Monitoring Web Information Sources. *Proc. of VLDB*, pp. 360–371, (2004).

[35] H. Roitman, A. Gal and L. Raschid. Capturing Approximated Data Delivery Tradeoffs. *ICDE* (2008).

[36] H. Roitman, A. Gal and L. Raschid. Satisfying Complex Data Needs using Pull-Based Online Monitoring of Volatile Data Sources. *ICDE*, 2008.

[37] H. Roitman. Profile Based Online Data Delivery - Model and Algorithms. *Ph.D. Thesis* (2008).

[38] H. Roitman, A. Gal, L. Raschid. On Trade-offs in Event Delivery Systems. The 4th ACM International Conference on Distributed Event-Based Systems (DEBS), 2010

[39] A. Schrijver. Combinatorial optimization: polyhedra and efficiency. Springer, 2003.

[40] J. Vondrák. Optimal approximation for the Submodular Welfare Problem in the value oracle model. *Proc. of ACM STOC*, 67–74, 2008.

[41] F. Zheng, S. Fung, W. Chan, F. Chin, C. Poon and P. Wong. Improved On-Line Broadcast Scheduling with Deadlines. *COCOON*, 320–329, 2006.

A Minimizing the Maximum Response Time - A Lower Bound

In this section, we consider the problem of minimizing the maximum response time and show that no randomized online algorithm can be $2-\epsilon$ competitive for any small constant $\epsilon > 0$ in the oblivious adversary model.

Lemma 6 *Consider the probabilistic experiment of throwing $m = rn$ balls, independently and uniformly, into n bins among which $h = \alpha n$ bins are colored red. Let Z be the number of empty red bins. The expectation of Z is given by $\mu = \mathrm{E}(Z) = \alpha n(1-\frac{1}{n})^m \sim \alpha n e^{-r}$. Furthermore, for any $\theta > 0, \epsilon > 0$, if n is sufficiently large, $\Pr(Z \geq (1+\theta)\mu) \leq \epsilon$.*

Theorem 8 *For any $\epsilon > 0$, there exists no randomized online algorithm for minimizing the maximum response time that is $(2-\epsilon)$-competitive in the oblivious adversary model.*

Proof: Let $\mathcal{P}$ be a probability distribution for choosing a request sequence ρ. For a deterministic online algorithm A, let its competitiveness under $\mathcal{P}$ be $\mathcal{C}_A^{\mathcal{P}}$, i.e, $\mathcal{C}_A^{\mathcal{P}} = \inf\{\mathcal{C} | E[cost_A(\rho)] \leq \mathcal{C} \cdot E[cost_O(\rho)]\}$ where $cost_A$ and $cost_O$ are the costs obtained by A and the offline optimal algorithm O, respectively. From Yao's Minmax Principle [32], we have

$$\inf_R \mathcal{C}_R = \sup_{\mathcal{P}} \inf_A \mathcal{C}_A^{\mathcal{P}}. \tag{8}$$

We will give a distribution $\mathcal{P}$ on request ρ and prove for any deterministic online algorithm A, $\mathrm{E}[\mathcal{C}_A^{\mathcal{P}}] \geq 2-\epsilon$ for any $\epsilon > 0$. Then, the theorem simply follows from (8).

The request sequence is simply formed as follows: There are $K+2$ phases where K is an integer and will be fixed later. Phase k consists of time slots $[(k-1)n+1, \ldots, kn]$, for $1 \leq k \leq K+2$. At the beginning of phase k (right before time slot $(k-1)n+1$), the client(adversary) requests a set of pages $P^k = \{p_1^k, p_2^k, \ldots, p_n^k\}$ then requests one randomly chosen page (with repetition) from P^k at each subsequent time slot during phase k. Note that the off-line optimal cost is n.

The high level idea of the analysis is very simple. We argue that after each phase the size of the backlog will increasing by a certain amount (according to a function f) with high probability until it approach to n.

Let c be some small universal constant between 0 and 1. Let $f(0) = 0$ and $f(k+1) = f(k) + c(1-f(k))^2$. It is not hard to see $\lim_{k\to\infty} f(k) = 1$. Let K be the constant such that $f(K) \geq 1-\delta$. Constants θ and δ are also chosen such that $(1-\theta)^K(2-\delta) \geq 2-\epsilon$.

We formally define the backlog B_t at time t as the set of requests not yet satisfied by the online algorithm A right after time t. We claim $|B_{kn}| \geq f(k) \cdot n$ holds with probability at least $(1-\theta)^k$ for all $k \geq 0$ if n is sufficiently large. It is not hard to see the maximum response time is at least the size of the backlog at any time. Therefore,

$$\mathrm{E}(\mathcal{C}_A^{\mathcal{P}}) \geq \frac{(1+f(K))\cdot n}{cost_O(\rho)} \cdot \Pr(|B_{Kn}| \geq (1+f(K))\cdot n)) \geq (2-\delta)(1-\theta)^K \geq 2-\epsilon.$$

Now, we prove the claim by induction on k. It is trivially true when $k=0$. Suppose the claim holds for k and we prove it holds for $k+1$. We first note that any reasonable online algorithm only broadcasts pages in B_{kn-1} once during phase k. We denote the set of time slots in phase k when A broadcasts pages in P^{k+1} by $T = \{t_1, t_2, \ldots, t_y\}, y \geq n - |B_{kn}|$. We write $T_1 = \{t_1, \ldots, t_{\lceil y/2 \rceil}\}$ and $T_2 = \{t_{\lceil y/2\rceil+1}, \ldots, t_y\}$. Let c' be another positive constant less than $1/2$ such that $(1-\theta)(\frac{1}{2}-c')\left(1-e^{-\frac{1}{2}}\right) \geq c$ and $c' \geq c$. We distinguish two cases.

Case 1: A broadcasts less than $y/2 - c'y$ distinct pages in T_1. In this case, it is easy to see the backlog will increase by at least $c'y$ at the end of phase k. By induction hypothesis,

$$|B_{(k+1)n}| \geq |B_{kn}| + c'y \geq |B_{kn}| + c'(n - |B_{kn}|) = n(f(k) + c' - c'f(k)) \geq f(k+1)n.$$

Case 2: A broadcasts at least $y/2 - c'y$ pages in T_1. Define the random variable

$$e_i = \begin{cases} 1, & p_i^k \text{ is broadcast by } A \text{ in } T_1 \text{ and requestedby the adversary in } T_2; \\ 0, & \text{Otherwise.} \end{cases}$$

It is easy to see $|B_{(k+1)n}| \geq |B_{kn}| + \sum_{i=1}^n e_i$. If we think of P^k as the bins, the pages broadcast by A in T_1 as red bins and each page requested in T_2 as balls, $\sum_i e_i$ is exactly the quantity of non-empty red bins.

From Lemma 6, we can see and

$$\Pr(\sum_i e_i \geq (1-\theta)\mu) \geq 1-\theta$$

where $\mu = E(\sum_i e_i) \geq y(\frac{1}{2}-c')(1-(1-\frac{1}{n})^{y/2})$. So, we can get that

$$\begin{aligned}
|B_{(k+1)}| &\geq |B_k| + \sum_i e_i \geq |B_k| + (1-\theta)\mu \\
&\geq |B_k| + (1-\theta)y\left(\frac{1}{2}-c'\right)\left(1-(1-\frac{1}{n})^{\frac{y}{2}}\right) \\
&\geq |B_k| + (1-\theta)(n-|B_k|)\left(\frac{1}{2}-c'\right)\cdot\left(1-(1-\frac{1}{n})^{\frac{(n-|B_k|)}{2}}\right) \\
&\geq |B_k| + (1-\theta)(\frac{1}{2}-c')\left(1-e^{-\frac{1}{2}}\right)n\left(1-\frac{|B_k|}{n}\right)^2 \\
&\geq f(k)n + cn(1-f(k))^2 = f(k+1)n
\end{aligned}$$

holds with probability at least $1-\theta$. The fourth inequality is due to $1-(1-\frac{1}{n})^{xn} \geq 1-e^{-x} \geq 1-(1-2(1-e^{-1/2})x)$ for $0 \geq x \geq 1/2$. and the last holds because $x + c(1-x)^2$ is monotonically increasing. This concludes case 2.

Therefore, no matter in which case, we have that

$$\begin{aligned}
\Pr\big(|B_{k+1}| \geq f(k+1)n\big) &\geq \Pr\big(|B_{k+1}| \geq f(k+1)n \wedge |B_k| \geq f(k)n\big) \\
&= \Pr\big(|B_{k+1}| \geq f(k+1)n \;\big|\; |B_k| \geq f(k)n\big)\Pr(|B_k| \geq f(k)n) \\
&\geq (1-\theta)^{k+1}.
\end{aligned}$$

B Omitted Proofs

Proof of Theorem 6 Let $\mathcal{P}_1$ and $\mathcal{P}_2$ are two disjoint set of pages such that $|\mathcal{P}_1| = s$ and $|\mathcal{P}_2| = N-s$. Consider the following two types of requests.

> `Type 1`: At each time t during $[0, N-s-1]$, $(M+1)$ requests for each page in $\mathcal{P}_1$ arrive, a total of $(M+1)s$ requests.
>
> `Type 2`: At time 0, M requests for each page in $\mathcal{P}_2$ arrive, a total of $M(N-s)$ requests.

All requests have the same deadline N. Here, N and M are such that $N \gg M \gg s$. Let $\mathcal{O}$ and $\mathcal{M}$ denote the requests which MAPF and OPT satisfy respectively. Note that MAPF is busy during $[0, N-s]$ processing only the requests for `Type 1`. Thus, MAPF can satisfy $(N-s)(M+1)s$ requests for `Type 1` and s^2M requests for `Type 2`, a total of $|\mathcal{M}| = (N-s)(M+1)s + s^2M$. On the other hand, we let OPT schedule the requests for `Type 2` during $[0, N-s]$ and those for `Type 1` during $[N-s, N]$, thereby satisfying all the requests, a total of $|\mathcal{O}| = (N-s)(M+1)s + M(N-s)$. It is easy to see that $|\mathcal{O}|/|\mathcal{M}|$ can be arbitrarily close to $(s+1)/s$ when $N \gg M \gg s$. □

Proof of Lemma 5 The proof for the first part is very similar to that of Lemma 2, and we omit the proof. To prove the second part, consider any request $J_{p,i}$. The fractional latency for the request $J_{p,i}$ is $\sum_{t\in\mathcal{T}_{p,i}}(t-a_{p,i})x_{p,i}^{(t)}$. Let $D(J_{p,i})$ be the latency of request $J_{p,i}$ in the integral schedule; it is 0 if the request is not captured. We note that due to step 2, a request $J_{p,i}$ may not be captured, even if p is broadcast within its window. In this case, no latency is incurred. The event $(D(J_{p,i}) = t - a_{p,i})$ occurs only when page p is broadcast at time t and the request is captured in the second step. Thus $\Pr[D(J_{p,i}) = t - a_{p,i}] \leq y_p^{(t)} \frac{x_{p,i}^{(t)}}{y_p^{(t)}} = x_{p,i}^{(t)}$; this bound also holds when $y_p^{(t)} = 0$, since $x_{p,i}^{(t)} \leq y_p^{(t)} = 0$. Thus we obtain $\mathrm{E}[D(J_{p,i})] = \sum_{t \in \mathcal{T}_{p,i}} (t - a_{p,i}) \Pr[D(J_{p,i}) = t - a_{p,i}] \leq \sum_{t \in \mathcal{T}_{p,i}} (t - a_{p,i}) x_{p,i}^{(t)}$. Summing up the expected latency over all requests completes the proof. $\square$