Better Scalable Algorithms for Broadcast Scheduling

NIKHIL BANSAL, Eindhoven University of Technology
RAVISHANKAR KRISHNASWAMY, Princeton University
VISWANATH NAGARAJAN, IBM T.J. Watson Research Center

In the classical *broadcast scheduling problem*, there are n pages stored at a server, and requests for these pages arrive over time. Whenever a page is broadcast, it satisfies all outstanding requests for that page. The objective is to minimize average *flowtime* of the requests. For any $\epsilon > 0$, we give a $(1+\epsilon)$-speed $O(1/\epsilon^3)$-competitive online algorithm for broadcast scheduling. This improves over the recent breakthrough result of [Im and Moseley 2010], where they obtained a $(1+\epsilon)$-speed $O(1/\epsilon^{11})$-competitive algorithm. Our algorithm and analysis are considerably simpler than [Im and Moseley 2010]. More importantly, our techniques also extend to the general setting of *non-uniform page-sizes* and *dependent-requests*. This is the first scalable algorithm for broadcast scheduling with varying size pages, and resolves the main open question from [Im and Moseley 2010].

Categories and Subject Descriptors: F.2.2 [**Nonnumerical Algorithms and Problems**]: Sequencing and Scheduling

General Terms: Algorithms

Additional Key Words and Phrases: Online Algorithms, Broadcast Scheduling, Resource Augmentation

ACM Reference Format:
Bansal, N., Krishnaswamy, R., and Nagarajan, V. Better scalable algorithms for broadcast scheduling. ACM Trans. Algor. 9, 4, Article 39 (March 2010), 24 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

We consider the classical problem of scheduling in a broadcast setting to minimize the average response time. In this problem, there are n pages, and requests for these pages arrive over time. There is a single server that can broadcast pages. Whenever a page is transmitted, it satisfies all the outstanding requests for that page. In the most basic version of the problem, we assume that time is slotted and that each page can be broadcast in a single time slot. Any request r is specified by its arrival time $a(r)$ and the page $p(r)$ that it requests; we let $[m] := \{1, 2, \ldots, m\}$ denote the set of all requests. A broadcast schedule is an assignment of pages to time slots. The flow-time (or response time) of request r under a broadcast schedule equals $b(r) - a(r)$ where $b(r) \geq a(r) + 1$ is the earliest time slot after $a(r)$ when page $p(r)$ is broadcast. The objective is to minimize the average flow-time, i.e. $\frac{1}{m} \cdot \sum_{r \in [m]} (b(r) - a(r))$. Note that the optimal value is at least one.

More general versions of the problem have also been studied. One generalization is to assume that pages have different sizes. A complicating issue in this case is that a request for a page may arrive in the midst of a transmission of this page. There are two

A preliminary version of this paper was published in ICALP 2010. This research is partially supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.022.211.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

natural models studied to handle this issue, depending on whether the client has the ability to cache the data or not. In the caching version, a request is considered satisfied as soon as it sees one complete transmission of a page, *regardless of the order* in which it receives the data (so it could first receive the latter half of the page and then receive the first half). Without a cache, a request can only be satisfied when it has received the contents of the page *in order*. The latter version is natural, for example, with movie transmissions, while the former is more natural for say data file transmissions. When pages have arbitrary sizes, it is also standard to consider preemptive schedules (i.e. transmission of a page need not occur at consecutive time-slots). This is because no reasonable guarantee can exist if preemption is disallowed.

Another generalization is the case of so called *dependent* requests. Here a request consists of a subset of pages, and this request is considered completed only when all the pages for this request have been broadcast.

1.1. Previous Work

The broadcast scheduling setting has been studied extensively in the last few years, both in the offline and online setting. Most of the work has been done on the most basic setting with unit page sizes and no dependencies. In addition to minimizing the average response time, various other metrics such maximum response time [Bartal and Muthukrishnan 2000; Chang et al. 2011; Chekuri et al. 2009b; Chekuri and Moseley 2009] throughput maximization [Charikar and Khuller 2006; Kim and Chwa 2004; Fung et al. 2008], delay-factor [Chekuri and Moseley 2009] etc. have also been studied quite extensively. We describe here the work related to minimizing the average response time. For the offline version of this problem, the first guarantee of any kind was a 3-speed, 3-approximation due to [Kalyanasundaram et al. 2000]. After a sequence of works [Gandhi et al. 2004; Gandhi et al. 2006; Bansal et al. 2005], an $O(\log^2 n/\log\log n)$-approximation based on iterated rounding techniques was obtained by [Bansal et al. 2008]. This is currently the best approximation known for the problem. It is also known that the problem is NP-Hard [Erlebach and Hall 2004; Chang et al. 2011]. While no APX-hardness result is known, it is known that the natural LP relaxation (which is the basis of all known results for this problem), has a (rather small) integrality gap of $28/27 = 1.037$ [Bansal et al. 2005].

In the online case, which is perhaps more interesting for practical applications of the problem, very strong lower bounds are known. In particular, any deterministic algorithm must be $\Omega(n)$ competitive and any randomized algorithm must be $\Omega(\sqrt{n})$ competitive [Kalyanasundaram et al. 2000; Bansal et al. 2005]. Thus, it is most natural to consider the problem in the resource augmentation setting, where the online algorithm is provided a slightly faster server than the optimum offline algorithm. The first positive result was due to [Edmonds and Pruhs 2003] who gave an algorithm B-Equi and showed that it is $(4+\epsilon)$-speed, $O(1/\epsilon)$-competitive. The algorithm B-Equi produced a schedule where several pages may be transmitted fractionally in a single time slot. [Edmonds and Pruhs 2003] also showed how to convert B-Equi into a valid schedule (i.e. only one page is transmitted in each time slot) using another $(1+\epsilon)$-speedup and losing a factor of $1/\epsilon$ in the competitive ratio, which gave a $(4+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive algorithm.

The result of [Edmonds and Pruhs 2003] is based on a very interesting idea. They related broadcast scheduling to another scheduling problem on multiprocessors known as non-clairvoyant scheduling with sublinear-nondecreasing speed-up curves. This problem is very interesting in its own right with several applications. It was introduced earlier by [Edmonds 2000] who gave a $(2+\epsilon)$-speed, $O(1/\epsilon)$-competitive algorithm called Equi for the non-clairvoyant scheduling problem. [Edmonds and Pruhs

2003] showed that the broadcast scheduling problem can be reduced to non-clairvoyant scheduling while losing a factor of 2 in the speed up required. Given the $(2+\epsilon)$-speed, $O(1/\epsilon)$-competitive algorithm Equi, this yields the $(4+\epsilon)$-speed, $O(1/\epsilon)$-algorithm B-Equi for broadcast (where pages are transmitted fractionally in each time-slot).

Recently, [Edmonds and Pruhs 2009] gave a very elegant algorithm called $\mathsf{LAPS}(\beta)$ for the non-clairvoyant scheduling problem mentioned above. They showed that for any $\epsilon > 0$, the algorithm $\mathsf{LAPS}(\epsilon/2)$ is $(1+\frac{\epsilon}{2})$-speed $O(1/\epsilon^2)$ competitive. Using the [Edmonds and Pruhs 2003] reduction from broadcast scheduling to non-clairvoyant scheduling mentioned above, this implies a $(2+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive 'fractional' broadcast schedule. Losing another factor of $1/\epsilon$, this can be converted to a valid broadcast schedule that is $(2+\epsilon)$-speed, and $O(1/\epsilon^3)$-competitive. These results [Edmonds and Pruhs 2003; Edmonds and Pruhs 2009] also hold when page sizes are non-unit but preemption is allowed.

Another natural online algorithm that has been studied is Longest Wait First (LWF). This is a natural greedy algorithm that at any time broadcast the page for which the total waiting time of outstanding requests is the highest. [Edmonds and Pruhs 2005] showed that LWF is 6-speed, $O(1)$-competitive. They also showed that no $n^{o(1)}$ guarantee is possible unless the speedup is at least $(1+\sqrt{5})/2 \approx 1.61$. In particular, this rules out the possibility of LWF being a $(1+\epsilon)$-speed, $O_\epsilon(1)$-competitive (scheduling algorithms with such guarantees are referred to as *fully scalable*). Recently, the results for LWF has been improved by [Chekuri et al. 2009a]. They show that LWF is 2.74-speed, $O(1)$-competitive. They also improve the lower bound on speed up required to $2-\epsilon$.

Until recently, a major open question in the area had been whether there are fully scalable algorithms. Intuitively, these algorithms are important from a practical point of view, since one would expect them to perform closest to an optimal algorithm in practice. See [Kalyanasundaram and Pruhs 2000; Pruhs et al. 2004] for a formal discussion of this issue. Recently, in a breakthrough result, [Im and Moseley 2010] obtained the first scalable algorithms for broadcast scheduling. In particular, they design an algorithm called $LA\text{-}W$, that is $(1+\epsilon)$-speed, $O(1/\epsilon^{11})$-competitive. This algorithm is similar to LWF, but it favors pages that have recent requests. The analysis of $LA\text{-}W$ is based on a rather complicated charging scheme. Additionally, the algorithm in [Im and Moseley 2010] only works for unit-size pages, and the authors leave open the question for varying-size pages.

The case of dependent requests has been studied by [Robert and Schabanel 2007]. They show that a generalization of the B-Equi algorithm, called B-EquiSet is $(4+\epsilon)$-speed, $O(1/\epsilon^3)$-competitive, even in the setting where pages have arbitrary lengths (with preemptions).

1.2. Our Results

In this paper we give fully scalable algorithms for broadcast scheduling with improved guarantees. Our algorithm and analysis are much simpler than that of [Im and Moseley 2010], and they also extend to the general setting with non-uniform page sizes and dependent requests. In particular we prove the following results:

THEOREM 1.1. *If all pages are of unit size, then for every $0 < \epsilon \leq 1$, there is a $(1+\epsilon)$-speed, $O\left(\frac{1}{\epsilon^2}\right)$-competitive randomized online algorithm for broadcast scheduling.*

We note that for the problem above, [Bansal et al. 2005] show a lower bound of $\Omega(\frac{1}{\epsilon})$ on the competitive ratio on any randomized algorithm, if a speedup of $1+\epsilon$ is allowed.

We also give a deterministic algorithm with a slightly worse guarantee.

THEOREM 1.2. *If all pages are of unit size, then for every $0 < \epsilon \leq 1$, there is a $(1+\epsilon)$-speed, $O\left(\frac{1}{\epsilon^3}\right)$-competitive deterministic online algorithm for broadcast scheduling.*

Our algorithm and its analysis are inspired by the algorithm LAPS for non-clairvoyant scheduling [Edmonds and Pruhs 2009]. Our main idea is to bypass the [Edmonds and Pruhs 2003] reduction (from broadcast scheduling to non-clairvoyant scheduling) that loses a factor of 2 in the speedup and directly adapt the ideas used in LAPS to the broadcast scheduling setting. To this end, we first consider the *fractional* version of the problem (i.e. pages can be fractionally transmitted in each time-slot) and show using a potential function argument, that a variant of LAPS (adapted to the broadcast setting) is $(1+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive. Then we show how to round this fractional schedule in an online manner to obtain an integral schedule (i.e. only one page transmitted in each time-slot). This idea of reducing broadcast scheduling to a fractional version, and solving the fractional version was also used implicitly in the algorithms of [Edmonds and Pruhs 2003; Edmonds and Pruhs 2005]. However we consider a different notion of fractional relaxation, which is crucial to obtain a fully scalable algorithm.

Our algorithm and the analysis can be extended to a more general setting where the pages have arbitrary sizes, and the requests have dependencies. In this problem, different pages have different (integer) sizes, and the requests arrive for *subsets* of pages. A request is satisfied only when it receives all the pages in its associated subset, and a request receives a page only if its contents have been broadcast in order, starting from the beginning, i.e., the request does not cache the page blocks. Also, to obtain any reasonable guarantees with arbitrary page-sizes, one needs to consider the preemptive version (we give an example in Section 5.3 for completeness). By preemptive we still mean that only one page is transmitted in each time-slot; however as pages have arbitrary sizes, the complete transmission of a page may involve several (possibly non-consecutive) time-slots. When all page-sizes are unit, a valid preemptive schedule in fact does not preempt any page.

In Section 5 we prove the following generalization of Theorem 1.2.

THEOREM 1.3. *Consider the broadcast scheduling setting where pages have arbitrary sizes and requests are dependent. Moreover, no cache is available. Then, if preemption is allowed, for every $0 < \epsilon \leq 1$, there is a $(1+\epsilon)$-speed, $O\left(\frac{1}{\epsilon^3}\right)$-competitive deterministic online algorithm.*

Thus we resolve the main open question from [Im and Moseley 2010], by obtaining a scalable algorithm for broadcast scheduling with varying page sizes. The approach here is similar to that for unit-size pages, namely reducing to fractional broadcast scheduling. However the rounding algorithm used to achieve this reduction is much more involved than for unit-sizes.

Remark: Our algorithm can be modified so that the amortized number of preemptions per page is $O(\log n)$. That is, if a schedule transmits k pages over the entire time horizon, then the number of preemptions is at most $O(k \log n)$.

Note that we state Theorem 1.3 only for the version where there is no cache available. In the setting where cache is available, the problem can be reduced to dependent requests with unit size pages, by replacing a page p of length ℓ_p by a dependent request consisting of the corresponding ℓ_p unit size pages.

COROLLARY 1.4. *For every $0 < \epsilon \leq 1$, there is a $(1+\epsilon)$-speed, $O\left(\frac{1}{\epsilon^3}\right)$-competitive deterministic online algorithm for broadcast scheduling with arbitrary size pages (and dependent requests) in the cache model.*

We remark that the potential function based analysis introduced here is quite versatile and has already been useful in other works such as [Edmonds et al. 2011; Gupta et al. 2010].

Connection to the non-clairvoyant scheduling problem. Given the results above, a natural question is whether the loss of factor 2 speed up in previous approaches [Edmonds and Pruhs 2003; Edmonds and Pruhs 2005] can be avoided in the reduction from broadcast scheduling to the non-clairvoyant scheduling problem mentioned above. It turns out that this is indeed possible. We give a reduction from fractional broadcast scheduling to non-clairvoyant scheduling that does not incur any loss in either the speed up or in the competitive ratio (i.e. it is a $(1, 1)$ transformation). The main idea to achieve this lies in the appropriate definition of the fractional broadcast problem, and the online rounding algorithms required to relate the broadcast problem to its fractional relaxation. As this reduction may be useful in other contexts, we present it for completeness in Section 6. Note that this reduction combined with the algorithm LAPS [Edmonds and Pruhs 2009] could also be used to prove our results. However, we have chosen to present our results directly without going via the non-clairvoyant reduction, since the proofs are simpler and cleaner this way.

Finally, in Section 7 we investigate an alternate variant of dependent requests, where a request is specified by several pages, but it is satisfied when any one of those pages is transmitted (instead of when all of these pages are transmitted). We show that this variant is much harder, even in the offline setting. In particular, any $n^{o(1)}$ approximation for the problem requires at least $\Omega(\log n)$ speed up (unless P=NP).

Roadmap. In Section 2, we present an online scalable algorithm for a fractional variant of the broadcast scheduling problem. We then present two online rounding techniques to obtain a scalable schedule for the original problem (with unit-size pages) in Sections 3 and 4. In Section 5, we generalize these results to obtain a scalable algorithm for the general case of broadcast scheduling with arbitrary page sizes and dependent requests. Then in Section 6, we present our ratio-preserving reduction from the broadcast scheduling problem to a well-known unicast scheduling problem (with jobs having sequential sections and parallelizable sections). Finally we show some lower bounds for other models in Section 7.

2. FRACTIONAL BROADCAST SCHEDULING

In this section we study a "fractional" variant of the broadcast scheduling problem and obtain a $(1 + \epsilon)$-speed, $O(1/\epsilon^2)$-competitive algorithm for it. Then in the subsequent two sections, we will show how to transform this algorithm into one for the (original) broadcast problem for the case of unit-size pages. To obtain the randomized algorithm in Section 4, we use an α-point randomized rounding technique from [Bansal et al. 2005]. To obtain the deterministic algorithm in Section 3, we present a different priority-based rounding technique which incurs an additional factor of $O(1/\epsilon)$ in the competitive ratio.

2.1. Problem Definition

The basic setting for the *fractional* broadcast scheduling problem is similar to the usual broadcast scheduling, namely a single server has n pages and requests for pages arrive online. The difference is that we work with continuous (instead of discrete) time, and the pages can be transmitted fractionally. At any continuous time instant t, a 1-speed schedule is allowed to broadcast each page $p \in [n]$ at *rate* $x_p(t)$, such that $\sum_{p=1}^{n} x_p(t) \leq 1$. In the resource augmentation setting, a feasible $(1+\epsilon)$-speed schedule means that $\sum_p x_p(t) \leq 1 + \epsilon$ at all times t. For any request $r \in [m]$, let us define its *completion time* under such a continuous schedule to be:

$$b(r) := \inf \left\{ s \ : \ \int_{a(r)}^{s} x_{p(r)}(t)dt \geq 1 \right\},$$

i.e. the time after the release of request r when one unit of page $p(r)$ has been broadcast. Finally the flow-time of request r equals $b(r) - a(r)$. Note that the flow-time of any request is at least one (for a 1-speed schedule). The objective in fractional broadcast scheduling is to compute a schedule that minimizes average flowtime, $\frac{1}{m}\sum_{r\in[m]}(b(r) - a(r))$.

2.2. Algorithm for Fractional Broadcast

At any continuous time t, let $N(t)$ denote the set of *active requests*, i.e. those which have not yet been completed. Let $N'(t)$ denote the $\lceil \epsilon |N(t)| \rceil$ "most-recent" requests among $N(t)$, i.e. those with the latest arrival times, with ties broken arbitrarily.

The algorithm time shares among the requests in $N'(t)$, i.e. the amount of page p transmitted at t is

$$x_p(t) := (1 + 4\epsilon) \cdot \frac{|\{r \in N'(t) : p(r) = p\}|}{|N'(t)|}, \quad \forall\, p \in [n].$$

Clearly, $\sum_{p=1}^{n} x_p(t) = 1 + 4\epsilon$ at all times t. For the sake of analysis, we also define:

$$y_r(t) := \begin{cases} \frac{1+4\epsilon}{|N'(t)|} & \text{if } r \in N'(t) \\ 0 & \text{if } r \notin N'(t) \end{cases}, \qquad \forall\, t \geq 0$$

In particular, $y_r(t)$ is the share of request of r if we distribute the processing power of $1 + 4\epsilon$ equally among requests in $N'(t)$. In the rest of this section, we prove the following.

THEOREM 2.1. *For any $0 < \epsilon \leq \frac{1}{4}$, the above algorithm is a $(1 + 4\epsilon)$-speed $O\left(\frac{1}{\epsilon^2}\right)$ competitive deterministic online algorithm for fractional broadcast scheduling.*

2.3. Analysis for Fractional Broadcast

Our analysis is based on a potential function argument inspired by that for LAPS [Edmonds and Pruhs 2009]. Let Opt denote an optimal fractional broadcast schedule for the given instance. Let On denote the fractional online schedule produced by the above algorithm. We will define a potential Φ and show that

$$\Delta\mathsf{On}(t) + \Delta\Phi(t) \leq \frac{2}{\epsilon^2}\Delta\mathsf{Opt}(t). \tag{2.1}$$

holds for (i) every infinitesimal intervals $[t, t + dt)$ such that no requests arrive or complete in On during this interval, and (ii) whenever new requests arrive at t or complete in On. Here $\Delta\mathsf{On}(t)$ (resp. $\Delta\mathsf{Opt}(t)$) denotes the cost incurred during $[t, t + dt)$ by the online (resp. offline) schedule. Let $N(t)$ (resp. $N^*(t)$) denote the number of active requests under On and Opt at time t. It is easy to see (by interchanging the order of summation in the objective function) that, during the interval $[t, t + dt)$, we have $\Delta\mathsf{On}(t) = N(t)dt$ and $\Delta\mathsf{Opt}(t) = N^*(t)dt$. For the case of request arrivals and completions in On, we assume that they are instantaneous and hence $\Delta\mathsf{On}(t) = \Delta\mathsf{Opt}(t) = 0$. Moreover, we will ensure that $\Phi(0) = \Phi(\infty) = 0$. By standard amortization arguments, inequality (2.1) would imply Theorem 2.1.

At any (continuous) time t and for any page $p \in [n]$, let $x_p^*(t)$ denote the rate at which Opt broadcasts p. We have $\sum_p x_p^*(t) \leq 1$ since the offline optimal is 1-speed. For page $p \in [n]$ and times $t_1 < t_2$, let $X(p, t_1, t_2) := \int_{t_1}^{t_2} x_p(t)dt$ denote the *(fractional) amount* *of page* p transmitted by On in the interval $[t_1, t_2]$. Likewise, $X^*(p, t_1, t_2) := \int_{t_1}^{t_2} x_p^*(t)dt$ denotes a similar quantity for the Opt schedule.

For any request $r \in [m]$, let $b^*(r)$ denote the completion time of r in Opt, and let $b(r)$ denote its completion time in On. For any $r \in [m]$, and times $t_1 < t_2$, define $Y(r, t_1, t_2) := \int_{t_1}^{t_2} y_r(t)dt$, i.e. the fractional time that the online algorithm has devoted towards *request* r in the interval $[t_1, t_2]$. As any request r is inactive after time $b(r)$, it holds that $y_r(t) = 0$ for all $t > b(r)$. Thus $Y(r, t, \infty) = Y(r, t, b(r))$ for all $r \in [m]$ and $t \geq 0$. Notice the difference that $Y(\cdot, \cdot, \cdot)$ is defined for requests while $X(\cdot, \cdot, \cdot)$ is defined for pages.

We now define the contribution of any request $r \in [m]$ to the potential as follows.

$$z_r(t) = Y(r, t, \infty) \cdot X^*(p(r), a(r), t)$$

Note that $z_r(t) \geq 0$ for any r and t. Intuitively, $z_r(t)$ captures how far the online algorithm lags behind the offline optimal, with respect to request r at time t. This is because $X^*(p(r), a(t), t) \geq 1$ if r is satisfied under Opt by time t, and $Y(r, t, \infty) = 0$ if r is already satisfied under On, as it will not be assigned any ”work” henceforth during (t, ∞).

Finally, the overall potential function is defined as:

$$\Phi(t) := \frac{1}{\epsilon} \cdot \sum_{r \in N(t)} \mathsf{rank}(r) \cdot z_r(t),$$

where rank is the function which orders active requests based on arrival times (with the highest rank of $|N(t)|$ going to the most recently arrived request that is still active and a rank of 1 to the earliest active request).

We now show that (2.1) holds.

Request Arrival: As $\Delta\mathsf{On} = \Delta\mathsf{Opt} = 0$ in this case, it suffices to show that $\Delta\Phi = 0$. When a request r arrives at time t, we have $z_r(t) = 0$ as r is entirely unsatisfied by Opt. Thus, Φ does not change due to r. Moreover, as the requests are ranked in the increasing order of their arrival, r gets the rank $N(t)+1$ and the ranks of other (active) requests are unaffected and hence $\Delta\Phi = 0$.

Request completes under Online Algorithm and leaves the set $N(t)$: As previously it suffices to show that $\Delta(\Phi) \leq 0$. When a request r leaves $N(t)$, by definition its $z_r(t)$ reaches 0 (because no work will be assigned to r henceforth and so $Y(r, t, \infty)$ will be 0). Moreover, when r leaves the set $N(t)$ the rank of the other requests $r' \in N(t)$ can only decrease. Since $z_{r'}(t) \geq 0$ for any r', the contribution due to these requests to the potential can only decrease. Thus $\Delta\Phi \leq 0$.

We now consider a sufficiently small interval $(t, t+dt)$ where neither of the above two events happen and show that (2.1) holds. There are two causes for change in potential:

Offline Opt broadcast in $(t, t+dt)$: We will show that the rate of change of Φ due to Opt working in this interval is $\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{Opt}} \leq \frac{1}{\epsilon}|N(t)|$. To see this, consider any page p. The rate at which page p is transmitted by Opt in this interval is $x_p^*(t)$ (by definition). This broadcast of page p causes the quantity $z_r(t)$ to increase for all those requests r with $p = p(r)$ that are alive in On at time t. To see the rates of their increases, let

$$C(t, p) := \{r \in [m] \mid p(r) = p,\ a(r) \leq t < b(r)\}$$

denote the active requests under On for page p at time t. As the rank of any alive request is at most $|N(t)|$, the total rate of increase in Φ over the interval $[t, t+dt)$ due to Opt's broadcast is at most:

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{Opt}} \le \frac{1}{\epsilon}|N(t)| \cdot \sum_{p=1}^{n} \sum_{r\in C(t,p)} Y(r,t,\infty)\cdot x_p^*(t). \tag{2.2}$$

We now show that $\sum_{r\in C(t,p)} Y(r,t,\infty) \le 1$ for any page p. Let $r' = \arg\max\{b(r) \mid r \in C(t,p)\}$ denote the request in $C(t,p)$ that is completed last by On. Since r' is active until $b(r')$ and $a(r') \le t$, it must be that On broadcasts at most 1 unit of page p during $[t, b(r')]$. As the total work assigned to requests r in $C(t,p)$ until $b(r')$ is no more than the amount of p transmitted until $b(r')$, and since r' is the last request to be completed in $C(t,p)$, this directly implies that

$$\sum_{r\in C(t,p)} Y(r,t,\infty) = \sum_{r\in C(t,p)} Y(r,t,b(r')) \le 1$$

Together with (2.2), we have that

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{Opt}} \le \frac{1}{\epsilon}|N(t)| \cdot \sum_{p=1}^{n} x_p^*(t) \le \frac{1}{\epsilon}|N(t)| \cdot, \tag{2.3}$$

where the last inequality uses that $\sum_p x_p^*(t) \le 1$ as Opt is a 1-speed algorithm.

Online broadcast in $(t, t+dt)$**:** Recall that On broadcasts page p at rate $x_p(t)$, and $y_r(t)$ is the rate at which On "works" on request r. Consider any fixed request $r \in N'(t) \setminus N^*(t)$, i.e. on which On works but has been completed by Opt. Observe that $X^*(p(r), a(r), t) \ge 1$ since Opt has completed request r by time t. Note also that $y_r(t) = (1+4\epsilon)/|N'(t)|$, i.e. $\frac{d}{dt}Y(r,t,\infty) = -(1+4\epsilon)/|N'(t)|$. Thus,

$$\frac{d}{dt}z_r(t) \;=\; X^*(p(r),a(r),t)\;\cdot\;\frac{d}{dt}Y(r,t,\infty) \;\le\; -\frac{1+4\epsilon}{|N'(t)|}, \qquad \text{for all } r \in N'(t)\setminus N^*(t).$$

Furthermore, since each request that On works on in $[t, t+dt)$ has rank at least $|N(t)| - |N'(t)| + 1 \ge (1-\epsilon)\cdot|N(t)|$, the potential Φ increases at rate,

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{On}} \quad\le\quad -\frac{1}{\epsilon}(1-\epsilon)\,|N(t)| \cdot \frac{(1+4\epsilon)}{|N'(t)|} \cdot (|N'(t)| - |N^*(t)|)\,.$$

Since $|N'(t)| \ge \epsilon|N(t)|$ and $(1-\epsilon)(1+4\epsilon) \ge (1+2\epsilon)$ for $\epsilon \le 1/4$, we get

$$\begin{aligned}\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{On}} &\le -\left(\frac{1}{\epsilon}+2\right)|N(t)| + \frac{1}{\epsilon^2}(1+4\epsilon)|N^*(t)| \\ &\le -\left(\frac{1}{\epsilon}+1\right)\cdot|N(t)| + \frac{2}{\epsilon^2}\cdot|N^*(t)|.\end{aligned} \tag{2.4}$$

Observe that $\frac{d}{dt}\mathsf{On}(t) = |N(t)|$ and $\frac{d}{dt}\mathsf{Opt}(t) = |N^*(t)|$. Using (2.3) and (2.4), we get that

$$\begin{aligned}\frac{d}{dt}\mathsf{On}(t) + \frac{d}{dt}\Phi(t) &\le |N(t)| + \frac{1}{\epsilon}|N(t)| - \left(\frac{1}{\epsilon}+1\right)|N(t)| + \frac{2}{\epsilon^2}|N^*(t)| \\ &\le \frac{2}{\epsilon^2}|N^*(t)| \;=\; \frac{2}{\epsilon^2}\cdot\frac{d}{dt}\mathsf{Opt}(t),\end{aligned}$$

which proves Equation (2.1). Thus by integrating from $t = 0$ to $t = \infty$ (and noting that $\Phi(0) = \Phi(\infty) = 0$), we obtain Theorem 2.1.

Note that the resulting fractional broadcast schedule may complete requests at fractional times. In the description of the rounding techniques, it will be useful to assume that we have a fractional broadcast schedule where each request arrives and completes at integral times. So we will just round up the fractional completion times to integer values — since any request incurs a flow time of at least one by definition, this can at most double the competitive ratio.

3. DETERMINISTIC ONLINE ALGORITHM

In this section, we show how to obtain an online deterministic (integral) broadcast schedule from the fractional schedule presented in Section 2.2. Our rounding technique requires an additional speed up of $(1+\epsilon)$, and loses an extra factor of $O(1/\epsilon)$ in the competitive ratio (as usual, $1+\epsilon$ speed up means that the algorithm gets to transmit one *additional free page* every $\lceil \frac{1}{\epsilon} \rceil$ time-steps). While our technique is similar to that used by [Edmonds and Pruhs 2003] to convert their "fractional" algorithm B-EQUI to B-EQUI-EDF, there are some crucial differences, as our notion of fractional schedule is different

3.1. Algorithm

Let On denote the fractional algorithm. Recall that $a(r)$ denotes the time a request r arrives, $b(r)$ denotes the time it is fractionally satisfied under On, and $x_p(t)$ denotes the fractional amount of page transmitted at time t. Let us define, the width $w(r)$ of request r as $w(r) = b(r) - a(r)$.

The rounding algorithm Rnd is a simple greedy algorithm. It maintains a queue $\mathcal{Q}$ (initially empty) of requests that are as yet unsatisfied by Rnd but have been fractionally satisfied by On. At any time, it transmits the request from the queue with the least width. The rounding algorithm at time t is given below.

ALGORITHM 1: OnlineRounding(t)

```
1: for any request $r$ that completes in On at time $t$, i.e. $b(r) = t$, and is yet unsatisfied under
   Rnd do
2:     enqueue the tuple $\langle r, w(r) = b(r) - a(r) \rangle$ into $\mathcal{Q}$.
3: end for
4: dequeue the request $\langle r_t, w(r_t) \rangle$ that has least width $w(r)$ among all elements in the queue
   $\mathcal{Q}$.
5: broadcast the page $p(r_t)$.
6: delete all requests $\langle r', w' \rangle$ in $\mathcal{Q}$ such that $p(r') = p(r)$.
7: repeat steps 4-6 again if $t$ is an integer multiple of $\lceil \frac{1}{2\epsilon} \rceil$.
```

3.2. Analysis

We will show that

THEOREM 3.1. *For any $0 < \epsilon < 1$ and $(1+\epsilon)$-speed fractional broadcast schedule* On*, the above algorithm produces a $(1+2\epsilon)$-speed integral schedule* Rnd *such that:*

$$\sum_r (b_I(r) - a(r)) \leq O\left(\frac{1}{\epsilon}\right) \cdot \sum_r (b(r) - a(r))$$

where $b_I(r)$ denotes the time r is satisfied in the integral schedule Rnd.

In fact we show the following stronger guarantee for every request. For any request r, the algorithm will broadcast page $p(r)$ during the interval $\left[a(r), b(r) + \frac{3}{\epsilon}(b(r) - a(r)) + 3\right]$.

Proof. Consider some request r. If there is a broadcast of the page $p(r)$ in the interval $[a(r), b(r)]$, then clearly the claimed bound for request r holds.

Therefore, let us assume that there has been no broadcast of page $p(r)$ in the interval $[a(r), b(r)]$. Since $p(r)$ is not broadcast during $[a(r), b(r)]$, it implies that the request r is added to the queue $\mathcal{Q}$ at time $t = b(r)$: as r is still unsatisfied at time $b(r)$. Let $w(r) = b(r) - a(r)$ be the width of request r. Also define t_ℓ to be the latest time before t when (i) a request of width greater than $w(r)$ was dequeued, or (ii) $\mathcal{Q}$ was empty, i.e.

$$t_\ell := \max\{z \leq t \mid \text{ at time } z\text{, either a request of width} > w(r) \text{ is dequeued, or } \mathcal{Q} = \emptyset\}$$

Clearly, by the greedy nature of the algorithm, at time t_ℓ there are no outstanding requests of width at most $w(r)$. Moreover during $[t_\ell+1, t]$, the algorithm always dequeues requests of width at most $w(r)$. We will show that there exists time $t' \leq t_\ell + \frac{3w(r)}{\epsilon} + 3$, at which there are no outstanding requests of width at most $w(r)$. In particular, this would mean that request r is dequeued before time t', i.e. $p(r)$ is broadcast during $[b(r), t_\ell + \frac{3w(r)}{\epsilon} + 3]$, which would complete the proof of the theorem.

Suppose, for the sake of contradiction that $\mathcal{Q}$ always has requests of width at most $w(r)$ during the entire interval $\mathcal{T} := [t_\ell + 1, t_\ell + \frac{3w(r)}{\epsilon} + 3]$. We first show the following claims about the fractional extent to which any page is broadcast during the time interval $\mathcal{T}$.

CLAIM 3.2. *Consider any page $p \in [n]$, and let t_1 and t_2 denote times (provided they exist) of some two successive broadcasts of p in $\mathcal{T}$. Then, in the fractional schedule $\int_{t_1}^{t_2} x_p(t)dt \geq 1$.*

Proof. As page p is broadcast at time t_2, it must have been initiated by some unsatisfied "trigger" request r' for p that was dequeued at time t_2. Furthermore, r' must have arrived after t_1 (i.e. $a(r') \geq t_1$) as otherwise, it would have been already serviced by the broadcast at t_1. Now, since it enters the queue by time t_2, it must be that $b(r') \leq t_2$, implying that $\int_{t_1}^{t_2} x_p(t)dt \geq \int_{a(r')}^{b(r')} x_p(t)dt \geq 1$. ■

CLAIM 3.3. *Consider any page $p \in [n]$ that is broadcast at least once during $\mathcal{T}$. If t_p denotes the time p was first broadcast in $\mathcal{T}$, then $\int_{t_\ell+1-w(r)}^{t_p} x_p(t)dt \geq 1$.*

Proof. By our assumption on $\mathcal{T}$, the algorithm only broadcasts requests having width at most $w(r)$ during $\mathcal{T}$. In particular, the "trigger request" r' that initiated the broadcast of p at time t_p must have width $b(r') - a(r') \leq w(r)$. Moreover, $b(r') \in [t_\ell + 1, t_p]$: indeed if $b(r') \leq t_\ell$, then the queue would have contained a request of width at most $w(r)$ at time t_ℓ, contradicting the definition of t_ℓ. This implies that $a(r') \geq b(r') - w(r) \geq t_\ell + 1 - w(r)$. Thus

$$\int_{t_\ell+1-w(r)}^{t_p} x_p(t)dt \geq \int_{a(r')}^{b(r')} x_p(t) \geq 1,$$

implying the claim. ■

Now, let N_p denote the number of broadcasts of a page p during the interval $\mathcal{T}$. Then, by the preceding two claims, we know that we can pack 1 unit of fractional broadcast (in On) of page p between (i) any two successive integral broadcasts of p in $\mathcal{T}$, and (ii) between time $t_\ell + 1 - w(r)$ until the first broadcast of p in $\mathcal{T}$. Therefore, we can pack at least N_p units of fractional broadcast of page p within the interval $[t_\ell + 1 - w(r), t_\ell] \cup \mathcal{T}$. Since On has $1 + \epsilon$ speed, $\sum_p N_p \leq (1 + \epsilon)(|\mathcal{T}| + w(r))$. On the other hand, as Rnd runs at speed $(1 + 2\epsilon)$ and $\mathcal{Q}$ is never empty during $\mathcal{T}$, we have $\sum_p N_p \geq (1 + 2\epsilon)|\mathcal{T}| - 1$.

These two bounds imply that $(1+\epsilon)\cdot(|\mathcal{T}|+w(r)) \geq (1+2\epsilon)|\mathcal{T}|-1$ which implies $|\mathcal{T}| \leq (1+\frac{1}{\epsilon})\cdot w(r)+\frac{1}{\epsilon} \leq 3w(r)/\epsilon$, contradicting our assumption that $\mathcal{T}$ has length $3w(r)/\epsilon+2$. ■

Clearly Theorem 3.1 combined with Theorem 2.1 implies Theorem 1.2.

4. RANDOMIZED ONLINE ALGORITHM

In this section, we give a randomized online procedure for rounding the fractional schedule into a valid (integral) schedule, using $1+\epsilon$ speedup. The advantage of this algorithm over the one in the previous section is that it only adds $O(1/\epsilon^2)$ in expectation to the response of a request (which can be subsumed in the competitive ratio). However, the drawback is that it assumes an oblivious adversary (which is unaware of outcomes of the randomization of the algorithm). The rounding algorithm is based on the α-point rounding technique. This result is originally from [Bansal et al. 2005]; we present it here for completeness.

The randomized online algorithm for broadcast scheduling works as follows. Consider some fractional schedule generated in an online manner, say by running On in Section 2. For notational convenience, we assume that On is running at speed 1 and obtain a $1+\epsilon$ speed integral schedule. The $1+\epsilon$ speed fractional schedule from Section 2 can be handled in an almost identical manner to give a $1+2\epsilon$ speed integral schedule.

Recall our notation that, for page $p \in [n]$ and times $t_1 < t_2$, $X(p,t_1,t_2) = \int_{t_1}^{t_2} x_p(t)dt$ denotes the (fractional) amount of page p broadcast in the interval $[t_1,t_2)$.

ALGORITHM 2: α-point rounding for broadcast

```
choose $\alpha_p \in [0,1)$ uniformly at random and independently, for each $p \in [n]$. This is done
initially, and the $\alpha_p$'s are fixed forever.
simulate the fractional online algorithm to obtain schedule On (Section 2).
for each integral time $t$ do
    enqueue into $\mathcal{Q}$ all pages $\{p \in [n] \mid \exists i \in \mathbb{Z}_+,\ X(p,0,t-1) < i+\alpha_p \leq X(p,0,t)\}$.
    dequeue the first page in $\mathcal{Q}$, and broadcast it. If $t$ is a multiple of $\lceil \frac{1}{\epsilon} \rceil$ perform this step
    twice.
end for
```

Recall that for any request $r \in [m]$, its arrival time is $a(r)$ and completion time under On is $b(r)$. The next claim is immediate from the α-point definition.

CLAIM 4.1. *For each request $r \in [m]$, the page $p(r)$ enters $\mathcal{Q}$ at some time during $[a(r),b(r)]$.*

Proof. Let $p := p(r)$ the page requested by r. Condition on any $\alpha_p \in [0,1)$. By definition of the fractional completion time of r, we have $X(p(r),a(r),b(r)) = 1$. So there exists some (fractional) time $t \in (a(r),b(r))$ such that $X(p(r),0,t) \in \alpha_p + \mathbb{Z}_+$. Since $a(r)$ and $b(r)$ are integral, the claim follows. ■

Next we bound the expected time spent by each page in the queue. First, the following lemma from [Bansal et al. 2005] shows that it suffices to consider the expected queue length at any time t.

LEMMA 4.2 ([BANSAL ET AL. 2005], LEMMA 3.1). *Consider some page p, and let t be some time when it is enqueued. Then the expected length of queue $\mathcal{Q}$ at time t (conditioned on p being enqueued at t), is at most 1 more than the (unconditional) expected queue length at t.*

Thus we bound the expected length of the queue $\mathcal{Q}$ at any time t.

LEMMA 4.3. *At any time t, the expected length of queue $\mathcal{Q}$ is at most $O(1/\epsilon^2)$.*

Proof. We follow the analysis in [Bansal et al. 2005]. $\mathcal{Q}_t$ denotes the queue length at time t. Fix a $k > \frac{3}{\epsilon^2}$; we will bound the probability $\Pr[\mathcal{Q}_t \geq 4k]$. Let t' be the latest time before t that $\mathcal{Q}$ is empty; note that t' is a random variable. For each $j \geq 0$, let η_j denote the event that $t' \in (t-(j+1)k, t-jk]$; observe that exactly one of the η_js occurs. So,

$$\Pr[\mathcal{Q}_t \geq 4k] \leq \sum_{j \geq 0} \Pr[(\mathcal{Q}_t \geq 4k) \wedge \eta_j]. \tag{4.5}$$

We now bound each of these terms.

CLAIM 4.4. *We have* $\Pr[(\mathcal{Q}_t \geq 4k) \wedge \eta_0] \leq e^{-k/2}$.

Proof. Observe that for $(\mathcal{Q}_t \geq 4k) \wedge \eta_0$ to happen, it must be that the number of enqueues during $[t-k, t]$ is at least $4k$ (denote this event H_0). We now upper bound $\Pr[H_0]$. For each $p \in [n]$ let $a_p = X(p, t-k, t)$, and random variable A_p denote the number of enqueues of page p during $[t-k, t]$. Since the α's for different pages are chosen independently, A_p's are independent r.v's. Additionally, by α-point rounding we have $A_p \in \{\lfloor a_p \rfloor, \lceil a_p \rceil\}$ for all $p \in [n]$; and $E[\sum_{p=1}^n A_p] = \sum_{p=1}^n a_p$. Also, we have $\sum_{p=1}^n a_p \leq k$. Event H_0 implies that $\sum_{p=1}^n A_p \geq 4k \geq 4 \cdot E\left[\sum_{p=1}^n A_p\right]$. Now using the multiplicative form of the Chernoff bound [Alon and Spencer 2000], $\Pr[H_0] \leq \exp(-k/2)$, and we obtain the claim. ■

CLAIM 4.5. *For each $j \geq 1$,* $\Pr[(\mathcal{Q}_t \geq 4k) \wedge \eta_j] \leq \exp(-\epsilon^2 jk/3)$.

Proof. For $(\mathcal{Q}_t \geq 4k) \wedge \eta_j$ to happen, it must be that the number of enqueues during $[t-jk-k, t]$ is at least $(1+\epsilon) \cdot jk + 4k$ (call this event H_j). This is because $\mathcal{Q}$ was empty at some time t' during $[t-jk-k, t-jk]$, the algorithm has speed $(1+\epsilon)$ and $\mathcal{Q}$ is never empty during $[t-jk, t]$. As in the previous claim, define the following. For each $p \in [n]$ let $a_p := X(p, t-jk-k, t)$, and random variable $A_p \in \{\lfloor a_p \rfloor, \lceil a_p \rceil\}$ denotes the number of enqueues of page p during $[t-jk-k, t]$. We also have $E[\sum_{p=1}^n A_p] = \sum_{p=1}^n a_p \leq (j+1)k$. Event H_j implies that:

$$\sum_{p=1}^n A_p \geq (1+\epsilon)jk + 4k \geq (1+\epsilon) \cdot E\left[\sum_{p=1}^n A_p\right].$$

Again by the Chernoff bound, $\Pr[H_j] \leq \exp(-\epsilon^2 jk/3)$. ■

Combining the two claims above with (4.5), we obtain:

$$\begin{aligned}
\Pr[\mathcal{Q}_t \geq 4k] &\leq e^{-k/2} + \sum_{j \geq 1} \exp(-\epsilon^2 jk/3) \\
&\leq e^{-k/2} + \exp(-\epsilon^2 k/3) \sum_{j=0}^{\infty} \left(\exp(-\epsilon^2 k/3)\right)^j \\
&\leq e^{-k/2} + 2 \cdot \exp(-\epsilon^2 k/3) \leq 3 \cdot \exp(-\epsilon^2 k/3),
\end{aligned}$$

where the second last inequality follows from $k \geq \frac{3}{\epsilon^2}$. Using this expression, we bound

$$E[\mathcal{Q}_t] = \sum_{\ell=0}^{\infty} \Pr[\mathcal{Q}_t > \ell] \leq \frac{12}{\epsilon^2} + 4 \sum_{k=3/\epsilon^2}^{\infty} \Pr[\mathcal{Q}_t > 4k] \leq \frac{12}{\epsilon^2} + 12 \sum_{k=3/\epsilon^2}^{\infty} e^{-\epsilon^2 k/3} \leq \frac{48}{\epsilon^2}.$$

This completes the proof of the lemma. ■

Using Claim 4.1 and Lemmas 4.3 and 4.2 we obtain that for each request $r \in [m]$, its *expected* flow-time in the integral schedule is at most $b(r) - a(r) + O(1/\epsilon^2)$. Since On is $O(1/\epsilon^2)$-competitive, the expected average flow time is at most $O(1/\epsilon^2)$ times the optimal.

Combined with Theorem 2.1 this proves Theorem 1.1.

5. THE GENERAL SETTING: DEPENDENT REQUESTS AND NON-UNIFORM PAGES

The non-uniform broadcast scheduling problem with dependencies is as follows. There are n pages with each page p having an integer size l_p; i.e. page p consists of l_p distinct blocks that are numbered 1 to l_p. Requests for *subsets* of these pages arrive over time. That is, a request r is specified by its arrival time $a(r)$ and a *subset* of pages $\mathcal{P}(r) \subseteq [n]$ that it requests; we let $[m]$ denote the set of all requests.

There is a single server that can broadcast one page-block per time slot. A broadcast schedule is an assignment of page-blocks (i.e. tuple $\langle p, i \rangle$ where $p \in [n]$ and $i \in \{1, \cdots, l_p\}$) to time slots. For any request r, page $p \in \mathcal{P}(r)$ is said to be *completed* if the server has broadcast after time $a(r)$, all the l_p blocks of page p in the order 1 through l_p. We consider a preemptive schedule and hence allow non-contiguous transmission of blocks. The flow-time of request r under a broadcast schedule equals $b(r) - a(r)$ where $b(r) \geq a(r) + 1$ is the earliest time slot after $a(r)$ when *all* the pages requested in $\mathcal{P}(r)$ have been completed. The objective is to minimize the average flow-time, i.e. $\frac{1}{m} \cdot \sum_{r \in [m]} (b(r) - a(r))$. We assume that the pages all have size at least 1, and therefore the optimal value is also at least one.

Our algorithm is again based on first solving the 'continuous' version of the problem, and then rounding this fractional schedule into a valid 'integral' schedule. Recall that an integral schedule is one where only one page is transmitted in each time slot; however since pages have arbitrary sizes, complete transmission of a page may occupy non-contiguous time-slots (i.e. preemptive schedule).

5.1. The Fractional Algorithm

In the fractional broadcast problem, the algorithm can transmit pages in a continuous manner. Here, at any (continuous) time instant t, the algorithm is allowed to broadcast each page $p \in [n]$ at rate $x_p(t)$, such that $\sum_{p=1}^{n} x_p(t) \leq 1$ for all t. Again in the resource augmentation setting, we allow $\sum_p x_p(t) \leq 1 + \epsilon$ for all t. For any request $r \in [m]$ and page $p \in \mathcal{P}(r)$, define

$$b(r,p) := \inf \left\{ s \ : \ \int_{a(r)}^{s} x_p(t) dt \geq l_p \right\},$$

i.e. the earliest time after the release of request r when l_p *units* of page p have been broadcast. The *completion time* of any request $r \in [m]$ is then:

$$b(r) := \max_{p \in \mathcal{P}(r)} b(r,p),$$

i.e. the time after the release of request r when all pages requested by r have been completely broadcast. Finally the flow-time of request r equals $b(r) - a(r)$. Note that in this fractional broadcast notion, we do not distinguish between the l_p distinct blocks of each page p; we only require the schedule to broadcast l_p units for page p (possibly out of order). The issue with the order of blocks will be handled in the rounding step later.

At any continuous time t, let $N(t)$ denote the set of *active requests*, i.e. those which have not yet been fractionally completed. Let $N'(t)$ denote the $\lceil \epsilon |N(t)| \rceil$ "most-recent" requests among $N(t)$, i.e. those with the latest arrival times. For each request $r \in N'(t)$, let $\mathsf{Unfin}(r, t)$ denote an arbitrary page $p \in \mathcal{P}(r)$ that has not been fractionally broadcast to an extent l_p since the arrival time $a(r)$. The algorithm then time shares among the pages $\{\mathsf{Unfin}(r,t) \mid r \in N'(t)\}$, i.e.

$$x_p(t) := (1+4\epsilon) \cdot \frac{|\{r \in N'(t) : \mathsf{Unfin}(r,t) = p\}|}{|N'(t)|}, \quad \forall\, p \in [n].$$

Clearly, $\sum_{p=1}^{n} x_p(t) \le 1 + 4\epsilon$ at all times t. For the sake of analysis, also define:

$$y_{r,p}(t) := \begin{cases} \frac{1+4\epsilon}{|N'(t)|} & \text{if } r \in N'(t), \text{ and } p = \mathsf{Unfin}(r,t) \\ 0 & \text{otherwise} \end{cases}, \qquad \forall\, t \ge 0$$

In particular, $y_{r,p}(t)$ is the share of request of r for page p, if we distribute the $1+4\epsilon$ processing equally among requests in $N'(t)$ and their pages.

5.2. Analysis of Fractional Broadcast

The analysis is very similar to that for the uniform broadcast scheduling case presented in Section 2.3. We first describe the potential function, and then use it to bound the competitive ratio.

We now revisit the notation used in the unit-size setting (and appropriately redefine some of them). Let Opt denote an optimal (offline) *fractional* broadcast schedule for the given instance, and let On denote the fractional online schedule produced by the above algorithm. For any request $r \in [m]$, let $b^*(r)$ denote the completion time of r in Opt, and let $b(r)$ denote its completion time in On. For any $r \in [m]$, page $p \in \mathcal{P}(r)$, and times $t_1 < t_2$, define $Y(r,p,t_1,t_2) := \int_{t_1}^{t_2} y_{r,p}(t)dt$ to denote the fractional time that the online algorithm has devoted towards *page p on behalf of request r* in the interval $[t_1,t_2]$ (recall that this generalizes the previous definition of $Y(\cdot,\cdot,\cdot)$ to the setting with dependent requests). Observe that for any request $r \in [m]$, page $p \in \mathcal{P}(r)$ and $t > b(r,p)$, we have $y_{r,p}(t) = 0$. Thus $Y(r,p,t,\infty) = Y(r,p,t,b(r,p))$ for all $r \in [m]$, $p \in \mathcal{P}(r)$, and $t \ge 0$.

At any (continuous) time t and for any page $p \in [n]$, let $x_p^*(t)$ denote the rate at which Opt broadcasts p. We have $\sum_p x_p^*(t) \le 1$ since the offline optimal is 1-speed. For page $p \in [n]$ and times $t_1 < t_2$, let $X^*(p,t_1,t_2) := \int_{t_1}^{t_2} x_p^*(t)dt$ denote the amount of page p transmitted by Opt in the interval $[t_1,t_2]$. At any continuous time t, let $N(t)$ and $N^*(t)$ denote the set of requests that are not completed in On and Opt respectively.

We now define the contribution of any request $r \in [m]$ and page $p \in \mathcal{P}(r)$ to the potential as follows.

$$z_{r,p}(t) \quad = \quad \frac{Y(r,p,t,\infty) \cdot X^*(p,a(r),t)}{l_p}$$

The total contribution of request r is then $z_r(t) = \sum_{p \in \mathcal{P}(r)} z_{r,p}(t)$. Note that $z_r(t) \ge 0$ for any r and t. Finally, the overall potential function is defined as

$$\Phi(t) \quad := \quad \frac{1}{\epsilon} \cdot \sum_{r \in N(t)} \mathsf{rank}(r) \cdot z_r(t),$$

where rank is the function which orders active requests based on arrival times (with the highest rank of $|N(t)|$ going to the request which arrived most recently and a rank of 1 to the oldest active request). The following analysis is almost identical to the one in Section 2.3, and is presented for the sake of completeness.

We will now show that the following inequality holds over all sufficiently small intervals $[t, t+dt)$ such that no requests arrive or complete in On during this interval.

Time instants where requests arrive or complete in On will be handled separately.

$$\Delta\mathsf{On}(t) + \Delta\Phi(t) \le \frac{2}{\epsilon^2}\Delta\mathsf{Opt}(t). \tag{5.6}$$

Since we ensure that $\Phi(0) = \Phi(\infty) = 0$, it is immediate to see that the total cost of the online algorithm is competitive with the optimal offline cost, up to a factor of $\frac{2}{\epsilon^2}$.

Request Arrival: We show that $\Delta\Phi = 0$ (clearly this suffices, since we can assume that arrivals happen instantaneously and hence $\Delta\mathsf{On} = \Delta\mathsf{Opt} = 0$). When a request r arrives at time t, we have $z_r(t) = 0$ as r is entirely unsatisfied by Opt. Thus, Φ does not change due to r. Moreover, as the requests are ranked in the increasing order of their arrival, the ranks of other requests are unaffected and hence $\Delta\Phi = 0$.

Request Completes under Online and leaves the set $N(t)$: When a request r leaves $N(t)$, by definition its $z_r(t)$ reaches 0. Moreover, the rank of any other request $r' \in N(t)$ can only decrease. Since $z_{r'}(t) \ge 0$ for any r', the contribution due to these requests to the potential can only decrease. Thus $\Delta\Phi \le 0$. And again, at that instant, $\Delta\mathsf{On} = \Delta\mathsf{Opt} = 0$, and hence equation (5.6) holds.

Now consider any sufficiently small interval $(t, t + dt)$ when neither of the above two events happen. There are two causes for change in potential:

Offline broadcast in $(t, t + dt)$: We will show that that rate of change of Φ due to Opt working in this interval is $\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{Opt}} \le \frac{1}{\epsilon}|N(t)|$. To see this, consider any page p. The rate at which the page is transmitted by Opt in this interval is $x^*_p(t)$. This broadcast of page p causes the quantity $z_{r,p}(t)$ to increase for all those requests r that are alive and have $p \in \mathcal{P}(r)$ unfinished in On at time t. Recall the definition of 'completion time' $b(r, p)$ for page p of request r. Define,

$$C(t, p) := \{r \in [m] \mid p \in \mathcal{P}(r),\ a(r) \le t < b(r, p)\}$$

Now, since the rank of any alive request is at most $|N(t)|$, we get that the rate of increase in Φ over the interval $[t, t + dt)$ due to Opt's broadcast is at most:

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{Opt}} \le \frac{1}{\epsilon}|N(t)| \cdot \sum_p \sum_{r\in C(t,p)} \frac{Y(r, p, t, \infty)x^*_p(t)}{l_p}. \tag{5.7}$$

We show that $\sum_{r\in C(t,p)} Y(r, p, t, \infty) \le l_p$ for any page p. The proof is exactly as in the unit-sized case. Let $r' = \arg\max\{b(r, p) \mid r \in C(t, p)\}$ be the request in $C(t, p)$ for which page p is completed last by On. Since page p for r' is *not* completed until $b(r', p)$ and $a(r') \le t$, it must be that On broadcasts at most l_p units of page p during $[t, b(r', p)]$; otherwise $b(r', p)$ would be smaller. Hence $\sum_{r\in C(t,p)} Y(r, pt, b(r', p)) \le l_p$. Observe that for all $r \in C(p, t)$ and $t \ge b(r', p)$, we have $y_{r,p}(t) = 0$ since $b(r, p) \le b(r', p)$. Thus $\sum_{r\in C(t,p)} Y(r, p, t, \infty) \le l_p$. Now plugging this into equation (5.7), we have that

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{Opt}} \le \frac{1}{\epsilon}|N(t)| \cdot \sum_p x^*_p(t) \le \frac{1}{\epsilon}|N(t)| \cdot dt \tag{5.8}$$

Recall that $\sum_p x^*_p(t) \le 1$ since Opt is 1-speed.

Online broadcast in $(t, t + dt)$: Recall that On broadcasts page p at rate $x_p(t)$, and $y_{r,p}(t)$ is the rate at which On works on page p for request r. Consider any fixed request $r \in N'(t) \setminus N^*(t)$, i.e. on which On works but is completed by Opt. Observe that for every $p \in \mathcal{P}(r)$, $X^*(p, a(r), t) \ge l_p$ since Opt has completed request r. Thus $z_r(t) \ge$

$\sum_{p\in\mathcal{P}(r)} Y(r,p,t,\infty)$. Note also that $\sum_{p\in\mathcal{P}(r)} y_{r,p}(t) = (1+4\epsilon)/|N'(t)|$. Thus,

$$\frac{d}{dt}z_r(t) \le -\sum_{p\in\mathcal{P}(r)} y_{r,p}(t) = -\frac{1+4\epsilon}{|N'(t)|}, \qquad \text{for all } r \in N'(t)\setminus N^*(t).$$

Furthermore, since each request that On works on in $[t, t+dt)$ has rank at least $|N(t)| - |N'(t)| + 1 \ge (1-\epsilon)\cdot|N(t)|$, the potential Φ increases at rate,

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{On}} \le -\frac{1}{\epsilon}(1-\epsilon)N(t)\cdot\frac{(1+4\epsilon)}{|N'(t)|}\left(|N'(t)| - |N^*(t)|\right).$$

Since $(1-\epsilon)(1+4\epsilon) \ge (1+2\epsilon)$ for $\epsilon \le 1/4$, we get

$$\left(\frac{d}{dt}\Phi(t)\right)_{\mathsf{On}} \le -\left(\frac{1}{\epsilon}+2\right)|N(t)| + \frac{1}{\epsilon^2}(1+4\epsilon)|N^*(t)| \le -\left(\frac{1}{\epsilon}+1\right)\cdot|N(t)| + \frac{2}{\epsilon^2}\cdot|N^*(t)|. \tag{5.9}$$

Observe that $\frac{d}{dt}\mathsf{On}(t) = |N(t)|$ and $\frac{d}{dt}\mathsf{Opt}(t) = |N^*(t)|$. Using (5.8) and (5.9),

$$\begin{aligned}\frac{d}{dt}\mathsf{On}(t) + \frac{d}{dt}\Phi(t) &\le |N(t)| + \frac{1}{\epsilon}|N(t)| - \left(\frac{1}{\epsilon}+1\right)|N(t)| + \frac{2}{\epsilon^2}|N^*(t)| \\ &\le \frac{2}{\epsilon^2}|N^*(t)| \;=\; \frac{2}{\epsilon^2}\cdot\frac{d}{dt}\mathsf{Opt}(t),\end{aligned}$$

which proves Equation (5.6). Thus we obtain:

THEOREM 5.1. *For any $0 < \epsilon \le \frac{1}{4}$, there is a $(1+4\epsilon)$-speed $O\left(\frac{1}{\epsilon^2}\right)$ competitive deterministic online algorithm for fractional broadcast scheduling with dependencies and non-uniform sizes.*

5.3. Deterministic Online Rounding of Fractional Broadcast

In this section, we focus on getting an integral broadcast schedule from the fractional schedule (in the no-cache model) in an online deterministic fashion. Given any 1-speed fractional broadcast schedule On, we will obtain a $(1+\epsilon)$-speed integral broadcast schedule Rnd (which gets to transmit an *additional unit of page* every $\lceil\frac{1}{\epsilon}\rceil$ time-steps) such that

$$\sum_r (b_I(r) - a(r)) \le O\left(\frac{1}{\epsilon}\right)\cdot\sum_r (b(r) - a(r))$$

where $b_I(r)$ (resp. $b(r)$) is the completion time of request r in the integral (resp. fractional) schedule. (This extends easily (as in Section 3) to an algorithm that transforms a $1+\epsilon$ speed fractional schedule to a $1+2\epsilon$ speed integral schedule.) An important issue in converting the fractional schedule to an integral one is that a valid broadcast of any page p now requires the l_p blocks of page p to be transmitted in the correct order. While this is relatively easy to guarantee if one is willing to lose a factor of 2 in the speed up, see for example the rounding step in [Edmonds and Pruhs 2003; Robert and Schabanel 2007], the algorithm here is much more subtle. The algorithm we present below is a (non-trivial) extension of that discussed in Section 3.

Algorithm Preliminaries. The rounding algorithm maintains the following items in its queue.

Jobs. For any request $r \in [m]$ and page $p \in \mathcal{P}(r)$, let *job* $\langle r,p\rangle$ denote the page p requested due to r. The arrival time of job $\langle r,p\rangle$ is the arrival time $a(r)$ of the corresponding request. We say that a job $\langle r,p\rangle$ is completed if the schedule contains a valid broadcast of page p starting after time $a(r)$. The completion time of job $\langle r, p\rangle$ in schedule Rnd (resp. On) is denoted $b_I(r, p)$ (resp. $b(r, p)$).

Tuples. The rounding algorithm maintains a queue of tuples (denoting transmissions of pages) of the form $\tau = \langle p, w, s, i\rangle$ where $p \in [n]$ is a page, $w \in \mathbb{R}_+$ is the *width*, $s \in \mathbb{Z}_+$ is the *start-time*, and $i \in \{1, \cdots, l_p\}$ is the *index* of the next block of page p to transmit. At each time-slot, the deterministic schedule broadcasts the current block of the tuple having *least width*.

Note the extension here from the scheme in Section 3; since page sizes are arbitrary, for each page we also track the time s when the current transmission began for this page, and an index that tracks the portion of this page that has been transmitted since time s.

ALGORITHM 3: GenRounding(t)

```
 1: initialize all jobs as unmarked when they arrive.
 2: simulate the fractional online algorithm to obtain schedule On.
 3: for any unmarked job $\langle r, p\rangle$ that completes under On at time $t$, i.e. $b(r, p) = t$, do
 4:     if there is a tuple $\tau = \langle p, w, s, i\rangle \in \mathcal{Q}$ of page $p$ with $s \geq a(r)$ then
 5:         update the width of tuple $\tau$ to $\min(w, b(r, p) - a(r))$.
 6:     else
 7:         insert new tuple $\langle p, b(r, p) - a(r), \infty, 1\rangle$ into $\mathcal{Q}$.
 8:     end if
 9: end for
10: dequeue the tuple $\tau = \langle p, w, s, i\rangle$ that has least width amongst all elements in $\mathcal{Q}$.
11: broadcast block $i$ of page $p$ in this time-slot.
12: if broadcast of $p$ corresponding to $\tau$ is just beginning (i.e. $i = 1$) then
13:     set $s = t$, i.e. equal to the current time slot .
14: end if
15: if broadcast of $p$ corresponding to $\tau$ is complete (i.e. $i = l_p$) then
16:     mark all jobs $\langle r', p\rangle$ of page $p$ having $a(r') \leq s$.
17: else
18:     enqueue the modified tuple $\langle p, w, s, i + 1\rangle$ into $\mathcal{Q}$.
19: end if
20: repeat steps 10-19 if $t$ is a multiple of $\lceil \frac{1}{\epsilon} \rceil$.
```

In order to bound the flowtime in schedule Rnd, we prove the following:

$$b_I(r, p) - b(r, p) \leq \frac{3}{\epsilon} \cdot \big(b(r, p) - a(r)\big) + \frac{5}{\epsilon}, \quad \text{for all jobs } \langle r, p\rangle \text{ .} \tag{5.10}$$

Note that this upper bounds $b_I(r, p) - a(r)$ with an additional $b(r, p) - a(r)$ term on the right hand side.

Consider any fixed job $\langle r, p\rangle$, and let $t = b(r, p)$. If at this time t, job $\langle r, p\rangle$ is *marked* then clearly $b_I(r, p) \leq t = b(r, p)$ and Equation (5.10) holds. So assume that $\langle r, p\rangle$ is unmarked. In this case (from the description of the algorithm) it must be that $\mathcal{Q}$ contains a tuple $\tau = \langle p, w, s, i\rangle$ where width $w \leq b(r, p) - a(r)$ and $s \geq a(r)$. Define,

$$t_A := \max\{z \leq t \mid \text{ at time } z\text{, either a request of width} > w \text{ is dequeued, or } \mathcal{Q} \text{ is empty}\}$$

$$t_B := \min\{z \geq t \mid \text{ at time } z\text{, either a request of width} > w \text{ is dequeued, or } \mathcal{Q} \text{ is empty}\}$$

Hence schedule Rnd always broadcasts some tuple of width at most w during interval $\mathcal{T} := (t_A, t_B)$, and there are no tuples of width at most w at times t_A and t_B. Clearly $b_I(r, p) \leq t_B$ and $b(r, p) = t \geq t_A$; so $b_I(r, p) - b(r, p) \leq t_B - t_A$ and it suffices to upper bound $t_B - t_A$ by the right hand side in (5.10).

Fix a page $q \in [n]$, and let Π_q denote the set of all tuples of page q that are broadcast (in even one time-slot) during $\mathcal{T}$. Let $N_q = |\Pi_q|$. We now prove some claims regarding Π_q.

CLAIM 5.2. *For each $\tau \in \Pi_q$, the start-time $s(\tau) \geq t_A - w$.*

Proof. Since τ is broadcast at some time-slot during $\mathcal{T}$, its width must be at most w at that time. Let $\langle r', q\rangle$ denote the job that caused τ's width to be at most w. Then it must be that $a(r') \leq s(\tau)$ and $b(r', q) \leq a(r') + w \leq s(\tau) + w$. Observe that at time t_A, queue $\mathcal{Q}$ contains no tuple of width at most w. Thus $b(r', q) \geq t_A$, i.e. $s(\tau) \geq t_A - w$, which proves the claim. ■

Based on this claim, we index tuples in Π_q as $\{\tau_j \mid 1 \leq j \leq N_q\}$ in increasing order of the start-times, i.e. $t_A - w \leq s(\tau_1) \leq s(\tau_2) \leq \cdots s(\tau_{N_q}) \leq t_B$. In the following, for page q and times $t_1 < t_2$, let $X(q, t_1, t_2)$ denote the amount of page q transmitted by fractional schedule On during interval (t_1, t_2).

CLAIM 5.3. *For any $1 \leq j \leq N_q - 1$, we have $X(q, s(\tau_j), s(\tau_{j+1})) \geq l_q$.*

Proof. Consider the time t' when tuple τ_{j+1} is first inserted into $\mathcal{Q}$. Since τ_j must have entered $\mathcal{Q}$ before τ_{j+1}, it must be that $s(\tau_j) < t' \leq s(\tau_{j+1})$; otherwise τ_{j+1} would not be inserted as a new tuple. Suppose τ_{j+1} is inserted due to the completion of job $\langle r', q\rangle$ in On. Then it must also be that $a(r') > s(\tau_j)$; otherwise job $\langle r', q\rangle$ would just have updated the width of τ_j and not inserted a new tuple. Clearly $b(r', q) = t'$, and hence $X(q, s(\tau_j), s(\tau_{j+1})) \geq X(q, a(r'), b(r', q)) \geq l_q$. ■

CLAIM 5.4. *$X(q, t_A - w, t_C) \geq l_q$, where $t_C = \max\{s(\tau_1), t_A + w\}$.*

Proof. Let $\langle r', q\rangle$ denote the *first* job that caused τ_1's width to be at most w (recall from Claim 5.2, there must be such a job). Again, it must be that $b(r', q) \geq t_A$ and so $a(r') \geq t_A - w$. We consider two cases:

(1) $s(\tau_1) \leq t_A$. In this case, we have $a(r') \leq s(\tau_1) \leq t_A$ and so $b(r', q) \leq a(r') + w \leq t_A + w$. Thus $X(q, t_A - w, t_A + w) \geq X(q, a(r'), b(r', q)) \geq l_q$.
(2) $s(\tau_1) > t_A$. Since start-time $s(\tau_1)$ of tuple τ_1 lies in $\mathcal{T}$, its width at time $s(\tau_1)$ is at most w. Hence $b(r', q) \leq s(\tau_1)$ for job $\langle r', q\rangle$. Thus in this case, $X(q, t_A - w, s(\tau_1)) \geq X(q, a(r'), b(r', q)) \geq l_q$.

Since $t_C = \max\{s(\tau_1), t_A + w\}$, the claim follows by the above cases. ■

Adding the expressions in Claims 5.3 and 5.4, we obtain:

$$\begin{aligned}
N_q \cdot l_q &\leq \sum_{j=1}^{N_q-1} X(q, s(\tau_j), s(\tau_{j+1})) + X(q, t_A - w, t_C) \\
&\leq X(q, t_A - w, s(\tau_1)) + \sum_{j=1}^{N_q-1} X(q, s(\tau_j), s(\tau_{j+1})) + X(q, t_A - w, t_A + w) \\
&= X(q, t_A - w, s(\tau_{N_q})) + X(q, t_A - w, t_A + w) \\
&\leq X(q, t_A - w, t_B) + X(q, t_A - w, t_A + w)
\end{aligned}$$

Now summing this inequality over all pages $q \in [n]$,

$$\sum_{q=1}^{n} N_q \cdot l_q \leq \sum_{q=1}^{n} X(q, t_A - w, t_B) + \sum_{q=1}^{n} X(q, t_A - w, t_A + w) \leq t_B - t_A + 3w + 2, \quad (5.11)$$

where the last inequality follows from the fact that On is 1-speed.

On the other hand, Rnd is always busy during $\mathcal{T}$: it is always broadcasting some tuple in $\bigcup_{q=1}^{n} \Pi_q$. Since Rnd has $1+\epsilon$ speed, we obtain:

$$\sum_{q=1}^{n} N_q \cdot l_q \geq (1+\epsilon) \cdot (t_B - t_A) - 3.$$

Combining this with (5.11), we have $t_B - t_A \leq \frac{3}{\epsilon} \cdot w + \frac{5}{\epsilon}$, which implies (5.10). Thus we obtain Theorem 1.3.

Remark: Our rounding algorithm can be modified so that the amortized number of preemptions per page is $O(\log n)$. That is, if a schedule transmits k pages over the entire time horizon, then the number of preemptions is at most $O(k \log n)$. To do this, recall that in the current algorithm if a page p begins transmission, then its width can only decrease over time until this page is completely transmitted. To guarantee logarithmic number of amortized preemptions, we can modify the algorithm so that it favors the transmission of the page it is currently transmitting and shifts to another page only if the width of that page is less than half the width of the current page. It can be shown that the number of preemptions decreases dramatically and the current analysis carries through with some minor modifications.

5.4. The Necessity of Preemption

We now give an example which illustrates the necessity of preemption in the case of non-uniform pages. In particular, we show that if preemption is disallowed, then for any arbitrarily large parameters b and c, there is an adversarial instance such that any online algorithm is at least b competitive even if it has a speed up of factor c.

Set $T = 16c^4b^2$ and $w = 2bcT^3$. Consider the following adversarial input: At time $t = 0$, there is one request for page p_0 of size $2cw$. Then, at each time slot iw for $i = 1, 2, 3, \ldots$, there are cbi requests for page p_i, where page p_i has size $\lfloor \frac{w}{cbi^3} \rfloor$. The adversary stops giving any requests when *(i)* the online algorithm completes broadcasting page p_0, or *(ii)* the index i reaches T.

In the first case, the broadcast of page p_0 must have spanned $2w$ consecutive time slots (even with c speed). Let $j \in \mathbb{Z}_+$ be the smallest index such that p_0 is being broadcast at time jw; note that p_0 is broadcast at least until time $jw + w$ (and at most till $jw + 2w$). Moreover, requests stop arriving after time $(j+1)w$. Note that requests for page p_j released at time jw wait for at least w time slots, incurring flow time at least $cbjw$. On the other hand, an adversary (that has 1-speed) could broadcast each page p_i (for $i = 1, \ldots, j+1$) as soon as its requests arrive at time iw; this is feasible since the size of each p_i is at most w. Then the adversary schedules page p_0 from time $(j+2)w$ to $(j+2+2c)w$. The cost it incurs would be at most:

$$\left(\sum_{i=1}^{j+1} cbi \cdot \frac{w}{cbi^3} \right) + (j+2+2c)w \quad \leq \quad (j+4+2c)w$$

which is at least $\Omega(b)$ times better than flow time of the online algorithm.

In the other case, if the online algorithm has not broadcast p_0 until time Tw, then its flow time is at least Tw. On the other hand we claim that there is a 1-speed offline solution with flow time at most $16c^4bw \leq \frac{1}{b} \cdot Tw$. Consider the solution that broadcasts p_0 in the first $2cw$ time-slots, and then the pages $\{1, \ldots, 2c\}$ that were released while p_0 was being broadcast. Since the sum of their sizes $\sum_{i=1}^{2c} (w/cbi^3) \leq \frac{2}{bc} w \leq w$, it follows that all the pages $p_1, p_2, \ldots, p_{2c}$ can be broadcast in the interval $[2cw, 2cw + w]$, right after completing p_0. Therefore, the requests corresponding to $p_1, p_2, \ldots, p_{2c}$ incur a flow time of at most $(2c+1)w$, and all subsequent requests (for pages $p_{2c+1}, p_{2c+2}, \ldots$) incur a collective waiting time of at most $\sum_{i=2c+1}^{T} cbi \cdot (w/cbi^3) \leq 2w$, since each of these pages can be broadcast immediately after its requests arrive. The cost of this offline solution is therefore at most $2cw + cb(1 + 2 + \ldots + 2c) \cdot (2c + 1)w + 2w \leq 16c^4bw$, which implies the claim.

6. BROADCAST SCHEDULING TO NON-CLAIRVOYANT UNICAST SCHEDULING

The non-clairvoyant unicast model (stated in a more general form in [Edmonds and Pruhs 2003]) is the following. The input is a set of n jobs that are to be executed on a single processor. The j^{th} job has the following parameters: an arrival time denoted by a_j, and a sequence of phases $\langle J_{j,1}, J_{j,2}, \ldots, J_{j,q_j} \rangle$. Each phase is an ordered pair $\langle w_{j,q}, \Gamma_{j,q} \rangle$ where $w_{j,q}$ denotes the amount of work and $\Gamma_{j,q}$ denotes its parallelizability (or the rate at which work is processed for that phase). That is, each phase can either be *fully parallel*, that is, a phase where $\Gamma(\beta) = \beta$, or *fully sequential*, that is, $\Gamma(\beta) = 1$ for every $\beta \in [0, 1]$. Here, β is the fraction of the processing power given to the job. Therefore, sequential work completes work at a rate of 1 even when absolutely no processing is allocated to it. Notice that we are only interested in these two extremities, although the original motivation for introducing speed-up curves was that different parts of code are parallelizable to different degrees.

A non-clairvoyant unicast scheduling algorithm is informed of the arrival of a new job j at time a_j, but is not aware of the nature of its phases (or the work to do in each phase). At each time instant t, it must partition the effective processing power between the jobs. All jobs begin in their first phase when they arrive. If a job j is executing a parallelizable phase q, it progresses from phase q to $q + 1$ at the first time t such that the total processing time allocated to j since the time it began phase q is at least w_q. On the other hand, if q is a completely sequential phase for j, the job stays in phase q for a duration of exactly w_q regardless of the amount of processing time the algorithm spends on j before moving to phase $q + 1$. The completion time of a job C_j is defined as the time at which the final phase of j completes. Its flow time is then, by definition, $C_j - a_j$. Also, for any job j, the non-clairvoyant algorithm is *only* notified of job arrival and completion, and not notified of which phase it is in or how long each phase is, etc.

In [Edmonds and Pruhs 2003], the authors show that the broadcast problem can be reduced to this non-clairvoyant unicast scheduling problem (in fact to the special case where each job has a sequential phase and at most one parallel phase), provided we have a factor 2 speedup. In the following, we show that if we care only about a *fractional broadcast* schedule (which can later be "rounded" online into an integer broadcast with $(1 + \epsilon)$-speedup), then we can avoid the loss of the factor 2.

The reduction is almost identical to the one in [Edmonds and Pruhs 2003], except for modifications that utilize our definition of fractional broadcast (that differs from [Edmonds and Pruhs 2003]). In the following, let $\mathcal{I}$ denote an instance of the online broadcast scheduling problem, and $\mathcal{A}$ be a deterministic non-clairvoyant algorithm for the "sequential-parallel unicast" problem. In Algorithm 4 we define $\mathcal{B}$, an online algorithm for the fractional broadcast problem which, using $\mathcal{A}$ as an oracle, decides which pages to broadcast at any time. In the process, we also define the instance $\mathcal{I}'$ for the unicast problem that $\mathcal{A}$ solves. Set $\delta \in (0, 1)$ to be an arbitrary fixed constant; this is required for technical reasons. We also assume that requests in $\mathcal{I}$ are numbered $1, 2, \ldots$ in order of their arrival.

We then show that the following inequalities hold.

$$\mathsf{Opt}(\mathcal{I}') \leq (1 + \delta) \cdot \mathsf{Opt}(\mathcal{I}) \tag{6.12}$$

$$\mathcal{B}(\mathcal{I}) \leq \mathcal{A}(\mathcal{I}') \tag{6.13}$$

ALGORITHM 4: Reducing fractional broadcast to non-clairvoyant scheduling

```
for each continuous time instant $t$ do
    for each request $r$ in $\mathcal{I}$ with $a(r) = t$ do
        create new job $j(r)$ for $\mathcal{I}'$ and inform $\mathcal{A}$ of its arrival.
    end for
    set $x_p(t) \leftarrow \sum_{r:p(r)=p} y_{j(r)}(t)$. Here $y$ denotes the unicast schedule output by $\mathcal{A}$ and $x$
    defines the broadcast schedule for $\mathcal{B}$.
    for each request $r$ in $\mathcal{I}$ with $\int_{a(r)}^{t} x_{p(r)}(\ell) d\ell = 1$ do
        set $C(r) \leftarrow t$, i.e. $r$ is completed in $\mathcal{I}$. Note that $j(r)$ is *not* yet completed in $\mathcal{I}'$.
    end for
    for each job $j(r)$ in $\mathcal{I}'$ with $\int_{C(r)}^{t} y_{j(r)}(\ell) d\ell = \frac{\delta}{2^r}$ do
        inform $\mathcal{A}$ that job $j(r)$ is completed.
    end for
end for
```

Above $\mathsf{Opt}(\mathcal{I})$ denotes the optimal *integral* broadcast schedule for $\mathcal{I}$. Notice that if $\mathcal{A}$ were an s-speed c-competitive algorithm for $\mathcal{I}'$, then we would get that $\mathcal{B}$ is an s-speed fractional broadcast that is $2c$-competitive w.r.t. the optimal integral broadcast. We now establish these inequalities.

To complete defining the instance $\mathcal{I}'$, we need to assign phases (and processing requirements) to each job. To this end, we compare $\mathsf{Opt}(\mathcal{I})$ to the schedule $\mathcal{B}(\mathcal{I})$ created by running our algorithm. For any request r in $\mathcal{I}$ let $C^*(r)$ denote its completion time under $\mathsf{Opt}(\mathcal{I})$; i.e. page $p(r)$ is broadcast in the interval $(C^*(r) - 1, C^*(r)]$. The job $j(r)$ in $\mathcal{I}'$ corresponding to request r in $\mathcal{I}$ is defined as follows:

— *Type 1 jobs.* If $C(r) < C^*(r) - 1$ then $j(r)$ has a sequential phase of duration $C(r) - a(r)$, followed by a parallel phase of work $\frac{\delta}{2^r}$.

— *Type 2 jobs.* If $C(r) \geq C^*(r) - 1$ then $j(r)$ has a sequential phase of duration $C^*(r) - a(r) - 1$ followed by a parallel phase with work $\int_{C^*(r)-1}^{C(r)} y_{j(r)}(t) dt + \frac{\delta}{2^r}$.

The following observation is immediate by the algorithm description.

OBSERVATION 6.1. *For any request r, its fractional completion time in $\mathcal{B}(\mathcal{I})$ is at most the completion time of the corresponding job $j(r)$ in the schedule created by $\mathcal{A}(\mathcal{I}')$.*

This is because, if a request r (in the broadcast instance $\mathcal{I}$) is fractionally completed by $\mathcal{B}$ at time $C(r)$, we declare completion of the corresponding job $j(r)$ in $\mathcal{A}$ only at the earliest time $t > C(r)$ when $\mathcal{A}$ schedules an *additional* $\frac{\delta}{2^r}$ parallel work of $j(r)$. Without this extra work, it is possible that $\mathcal{A}$ finishes a type 2 job $j(r)$ before $\mathcal{B}$ finishes r: the parallel work of $\int_{C^*(r)-1}^{C(r)} y_{j(r)}(t) dt$ in $j(r)$ may be completed strictly before $C(r)$ since r can be satisfied by other requests which correspond to the same page $p(r)$. The above observation immediately gives us inequality 6.13, and we now turn our attention to proving that equation 6.12 holds.

LEMMA 6.2. *Let $\mathsf{Opt}(\mathcal{I})$ be any optimal integral schedule for the broadcast instance. Then there exists a schedule $\mathsf{Sch}(\mathcal{I}')$ for the unicast instance such that, for any request r, the flow time of job $j(r)$ in $\mathsf{Sch}(\mathcal{I}')$ is at most $(1 + \delta)$ times the flow time incurred by r in $\mathsf{Opt}(\mathcal{I})$. Thus $\mathsf{Opt}(\mathcal{I}') \leq (1 + \delta) \cdot \mathsf{Opt}(\mathcal{I})$.*

Proof. We create schedule $\mathsf{Sch}(\mathcal{I}')$ for $\mathcal{I}'$ that for each integral time slot $(t - 1, t]$ does the following. Let p denote the page broadcast by $\mathsf{Opt}(\mathcal{I})$ during $(t - 1, t]$ and $C(t, p)$ the set of outstanding requests in $\mathcal{I}$ that were satisfied by this broadcast of page p. Define $\mathcal{W}_t$ to contain for each $r \in C(t, p)$, all the parallel work of $j(r)$, which is:

— $\int_{C^*(r)-1}^{C(r)} y_{j(r)}(\ell)d\ell + \frac{\delta}{2^r}$ units if job $j(r)$ is type 2, and
— $\frac{\delta}{2^r}$ units if $j(r)$ is type 1.

Let $1+\Delta_t$ (where $\Delta_t \geq 0$) denote an upper bound on the total work in $\mathcal{W}_t$. The final schedule $\mathsf{Sch}(\mathcal{I}')$ is obtained by assigning each $\mathcal{W}_t$ to the interval $(t-1+\sum_{\ell=0}^{t-1}\Delta_\ell,\ t+\sum_{\ell=0}^{t}\Delta_\ell]$. Note that this is a feasible work assignment since $|\mathcal{W}_t| \leq 1+\Delta_t$ for each t. Moreover, $\mathcal{W}_t$ performs the parallel work of all jobs $\{j(r) : r \in C(t,p)\}$ after their sequential phase (which ends by time $t-1$) and so $\mathsf{Sch}(\mathcal{I}')$ completes these jobs by the end of $\mathcal{W}_t$, i.e. at time $t+\sum_{\ell=0}^{t}\Delta_\ell$. Thus, the completion time of any job $j(r)$ in $\mathsf{Sch}(\mathcal{I}')$ is at most $C_r^* + \sum_{\ell=0}^{t}\Delta_\ell$. Below we show that:

$$\sum_{t\geq 0}\Delta_t \quad \leq \quad \delta. \tag{6.14}$$

This suffices to prove the lemma since the flow time of each job $j(r)$ in $\mathsf{Sch}(\mathcal{I}')$ can then be bounded by $C_r^* - a_r + \delta$ which is at most $(1+\delta)$ times the flowtime of r in $\mathsf{Opt}(\mathcal{I})$.

It only remains to prove (6.14). Consider any time slot $(t-1,t]$ where $\mathsf{Opt}(\mathcal{I})$ broadcasts page p, and which results in work $\mathcal{W}_t$ as above. For any $r \in C(t,p)$, let $L_r := \int_{C^*(r)-1}^{C(r)} y_{j(r)}(\ell)d\ell = \int_{t-1}^{C(r)} y_{j(r)}(\ell)d\ell$. We claim that $\sum_{r\in C(t,p)} L_r \leq 1$. Let request $r' := \arg\max_{r\in C(t,p)} C(r)$. Since r' is alive during $[t-1, C(r'))$, it must be that at most one unit of page p is broadcast by $\mathcal{B}$ during $[t-1, C(r'))$. In other words, $\sum_{r\in C(t,p)} \int_{t-1}^{C(r')} y_{j(r)}(\ell)d\ell < 1$. For all $r \in C(t,p)$, since $C(r) \leq C(r')$ we obtain $L_r \leq \int_{t-1}^{C(r')} y_{j(r)}(\ell)d\ell$, which implies $\sum_{r\in C(t,p)} L_r \leq 1$ as desired. Notice that $|\mathcal{W}_t| = \sum_{r\in C(t,p)} \left(L_r + \frac{\delta}{2^r}\right) \leq 1 + \sum_{r\in C(t,p)} \frac{\delta}{2^r}$. So we can set $\Delta_t = \sum_{r\in C(t,p)} \frac{\delta}{2^r}$, and

$$\sum_{t\geq 0}\Delta_t \quad = \quad \sum_{t\geq 0}\sum_{r\in C(t,p)} \frac{\delta}{2^r} \quad = \quad \sum_r \frac{\delta}{2^r} \quad \leq \quad \delta,$$

which proves (6.14) and also the lemma. ■

Thus we have proved:

THEOREM 6.3. *If there is a non-clairvoyant s-speed c-competitive deterministic algorithm for unicast scheduling, then there is an s-speed $2c$-competitive (w.r.t. an optimum integral schedule) algorithm for fractional broadcast scheduling.*

Combining this reduction with the $(1+\epsilon)$-speed $O(1/\epsilon^2)$-competitive online algorithm LAPS for unicast scheduling [Edmonds and Pruhs 2009], and the online rounding algorithm for fractional broadcast (Theorem 3.1), we obtain an alternate proof of Theorem 1.2.

7. BROADCAST SCHEDULING WITH DISJUNCTIVE REQUIREMENTS

In this section, we consider another generalization (disjunctive broadcast scheduling) of the usual broadcast problem, where each request r corresponds to a subset $S(r)$ of pages and a request is satisfied when *any* of the pages in $S(r)$ is broadcast. This is different from broadcast scheduling with dependencies [Robert and Schabanel 2007] since the request's requirement is a disjunction of page-broadcasts, as opposed to a conjunction. We observe that (assuming P≠NP) the offline version of this problem admits no sub-polynomial approximation guarantee unless the algorithm is allowed a speed-up of $\Omega(\log n)$.

THEOREM 7.1. *If there is an $o(m^{1/3})$-approximation algorithm for disjunctive broadcast scheduling with ρ speed-up, then there is a 4ρ-approximation algorithm for set-cover. Here m denotes the number of requests.*

Proof. The proof is a simple reduction from set-cover. Let $\mathcal{I}$ denote an instance of set-cover with universe $[N]$ and sets $\{A_i \subseteq [N]\}_{i=1}^{M}$. We construct an instance $\mathcal{J}$ of disjunctive broadcast scheduling using $T := N^2$ disjoint 'copies' of instance $\mathcal{I}$ as follows. There are $n := M \cdot T$ pages denoted $\{A_i^j \mid i \in [M], j \in [T]\}$ and $m := N \cdot T$ requests denoted $\{r_k^j \mid k \in [N],\ j \in [T]\}$. For each $j \in [T]$ and $k \in [N]$, we set $S(r_k^j) := \{A_i^j \mid k \in A_i, i \in [M]\}$. Note that requests and pages naturally correspond to T disjoint instances of $\mathcal{I}$.

Let $\kappa \in \{1, \cdots, M\}$ be a guess of the optimal set-cover value for $\mathcal{I}$ (we will try all values). The arrival times of the requests are then: $a(r_k^j) = (j-1) \cdot \kappa$ for all $k \in [N]$ and $j \in [T]$. Note that there is a 1-speed schedule for $\mathcal{J}$ (using the optimal set-cover for $\mathcal{I}$) having average flow-time at most κ. Suppose that there is some $o(m^{1/3})$-approximation for disjunctive broadcast scheduling with speed-up ρ. Since $\frac{T}{9N} = O(m^{1/3})$, this is also a $\frac{T}{9N}$-approximation. Let β denote the resulting schedule for instance $\mathcal{J}$; we now show how this implies a small set-cover for $\mathcal{I}$. Consider the first κT time slots, and let B denote the set of pages broadcast by β during these. Since β is ρ-speed, we have $|B| \leq \rho\kappa T$. For each $j \in \{1, \cdots, T/2\}$, define $B_j := \{i \in [N] \mid A_i^j \in B\}$. Let $T' \subseteq \{1, \cdots, T/2\}$ denote the indices $j \leq T/2$ such that $|B_j| \leq 4\rho\kappa$. Clearly $|T'| \geq T/4$. We claim that one of $\{B_j \mid j \in T'\}$ is a set-cover for $\mathcal{I}$. Suppose (for a contradiction) that this is not the case. Then, for each $j \in T'$ there is at least one request r_k^j (some $k \in [N]$) that is unsatisfied until time κT; since $j \leq T/2$ this request r_k^j has flow-time at least $\kappa T/2$. Thus the average flow-time of schedule β is at least $\frac{1}{NT} \cdot |T'|\kappa T/2 \geq \frac{\kappa T}{8N}$. However this contradicts the fact that schedule β is a $\frac{T}{9N}$-approximation. Since each B_j (for $j \in T'$) has size at most $4\rho\kappa$, we obtain a set-cover for $\mathcal{I}$ that is a 4ρ-approximation. ■

REFERENCES

ALON, N. AND SPENCER, J. H. 2000. *The probabilistic method* 2 Ed. Wiley, New York.

BANSAL, N., CHARIKAR, M., KHANNA, S., AND NAOR, J. 2005. Approximating the average response time in broadcast scheduling. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 215–221.

BANSAL, N., COPPERSMITH, D., AND SVIRIDENKO, M. 2008. Improved approximation algorithms for broadcast scheduling. *SIAM J. Comput. 38,* 3, 1157–1174.

BARTAL, Y. AND MUTHUKRISHNAN, S. 2000. Minimizing maximum response time in scheduling broadcasts. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 558–559.

CHANG, J., ERLEBACH, T., GAILIS, R., AND KHULLER, S. 2011. Broadcast scheduling: Algorithms and complexity. *ACM Transactions on Algorithms 7,* 4.

CHARIKAR, M. AND KHULLER, S. 2006. A robust maximum completion time measure for scheduling. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 324–333.

CHEKURI, C., IM, S., AND MOSELEY, B. 2009a. Longest wait first for broadcast scheduling. In *Proceedings of the Workshop on Approximation and Online Algorithms (WAOA)*. 62–74.

CHEKURI, C., IM, S., AND MOSELEY, B. 2009b. Minimizing maximum response time and delay factor in broadcast scheduling. In *Proceedings of the European Symposium on Algorithms (ESA)*. 444–455.

CHEKURI, C. AND MOSELEY, B. 2009. Online scheduling to minimize the maximum delay factor. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1116–1125.

EDMONDS, J. 2000. Scheduling in the dark. *Theor. Comput. Sci. 235,* 1, 109–141.

EDMONDS, J., IM, S., AND MOSELEY, B. 2011. Online scalable scheduling for the ℓ_k-norms of flow time without conservation of work. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 109–119.

EDMONDS, J. AND PRUHS, K. 2003. Multicast pull scheduling: When fairness is fine. *Algorithmica 36,* 3, 315–330.

EDMONDS, J. AND PRUHS, K. 2005. A maiden analysis of longest wait first. *ACM Transactions on Algorithms 1,* 1, 14–32.

EDMONDS, J. AND PRUHS, K. 2009. Scalably scheduling processes with arbitrary speedup curves. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 685–692.

ERLEBACH, T. AND HALL, A. 2004. NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *J. Scheduling 7,* 3, 223–241.

FUNG, S. P. Y., ZHENG, F., CHAN, W.-T., CHIN, F. Y. L., POON, C. K., AND WONG, P. W. H. 2008. Improved on-line broadcast scheduling with deadlines. *J. Scheduling 11,* 4, 299–308.

GANDHI, R., KHULLER, S., KIM, Y. A., AND WAN, Y.-C. J. 2004. Algorithms for minimizing response time in broadcast scheduling. *Algorithmica 38,* 4, 597–608.

GANDHI, R., KHULLER, S., PARTHASARATHY, S., AND SRINIVASAN, A. 2006. Dependent rounding and its applications to approximation algorithms. *J. ACM 53,* 3, 324–360.

GUPTA, A., IM, S., KRISHNASWAMY, R., MOSELEY, B., AND PRUHS, K. 2010. Scheduling jobs with varying parallelizability to reduce variance. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 11–20.

IM, S. AND MOSELEY, B. 2010. An online scalable algorithm for average flow time in broadcast scheduling. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1322–1333.

KALYANASUNDARAM, B. AND PRUHS, K. 2000. Speed is as powerful as clairvoyance. *J. ACM 47,* 4, 617–643.

KALYANASUNDARAM, B., PRUHS, K., AND VELAUTHAPILLAI, M. 2000. Scheduling broadcasts in wireless networks. In *Proceedings of the European Symposium on Algorithms (ESA)*. 290–301.

KIM, J.-H. AND CHWA, K.-Y. 2004. Scheduling broadcasts with deadlines. *Theor. Comput. Sci. 325,* 3, 479–488.

PRUHS, K., SGALL, J., AND TORNG, E. 2004. Online scheduling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J. Y.-T. Leung, Ed. CRC Press.

ROBERT, J. AND SCHABANEL, N. 2007. Pull-based data broadcast with dependencies: be fair to users, not to items. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 238–247.

Received ...; revised ...; accepted ...