Better Scalable Algorithms for Broadcast Scheduling

Nikhil Bansal* **Ravishankar Krishnaswamy**†
Viswanath Nagarajan*

November 2009
CMU-CS-09-174

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*IBM T.J. Watson Research Center.
†School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Work done while the author was visiting IBM T.J. Watson Research Center.

Keywords: Broadcast Scheduling, Online Algorithms

Abstract

In the classic *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 [IM10], where they obtained a $(1+\epsilon)$-speed $O(1/\epsilon^{11})$-competitive algorithm. Our algorithm and analysis are considerably simpler than [IM10]. 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 [IM10].

1 Introduction

We consider the classic problem of scheduling in a broadcast setting to minimize the average response time. In the broadcast scheduling 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 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 $[m]$ 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 natural models studied here, with caching and without caching. In the caching version, a request is considered satisfied as soon as it sees one complete transmission of a page (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 starts receiving the page from the beginning and when the page is completely transmitted. 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 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 [BM00, CEHK08, CIM09a, CM09] throughput maximization [CK06, KC04, ZFCCPW06], delay-factor [CM09] etc. have also been studied quite extensively. We describe here the work related to minimizing the average response time. We first consider the offline case. The first guarantee of any kind was a 3-speed, 3-approximation due to Kalyanasundaram, Pruhs and Veluthapillai [KPV00]. After a sequence of works [GKKW04, GKPS06, BCKN05], an $O(\log^2 n / \log \log n)$-approximation based on iterated rounding techniques was obtained by Bansal, Coppersmith and Sviridenko [BCS06]. This is currently the best approximation known for the problem. It is also known that the problem is NP-Hard [EH02, CEHK08]. While no APX-hardness result is known, it is known that the natural LP formulation (which is the basis of all known results for this problem), has a (rather small) integrality gap of $28/27 = 1.037$ [BCKN05].

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 [KPV00, BCKN05]. 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 [EP03] 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 [EP03] 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.

This result is based on a very interesting idea. The authors make a connection with 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 and was introduced in a previous paper by Edmonds [E00]. In that paper, Edmonds gave a $(2+\epsilon)$-speed, $O(1/\epsilon)$-competitive algorithm called Equi for the non-clairvoyant scheduling problem. Edmonds and Pruhs showed that the broadcast scheduling problem can be reduced to non-clairvoyant scheduling problem while losing a factor of 2 in the speed up required [EP03]. 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 [EP09] gave a very elegant algorithm called $\mathsf{LAPS}(\beta)$ for the non-clairvoyant scheduling problem. 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-Pruhs reduction from broadcast to non-clairvoyant scheduling mentioned above [EP03], this implies an $(2+\epsilon)$-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 [EP03, EP09] 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 [EP05] 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, $O(1)$-competitive, aka fully scalable, algorithm. Recently, the results for LWF has been improved by [CIM09b]. They show that LWF if 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, fully scalable are important from a practical point of view, since one would expect them to perform close to optimum in practice. See [KP00, PST04] for a formal discussion of this issue. Recently, in a breakthrough result, Sungjin Im and Ben Moseley [IM10] obtained the first scalable algorithms for broadcast scheduling. In particular, they design an algorithm call $LA-W$, 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 $LA-W$ is based on a rather complicated charging scheme. Additionally, the algorithm in [IM10] 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 [RS07]. 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 [IM10], 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 there is a lower bound of $\Omega(\frac{1}{\epsilon})$ on the competitiveness of even randomized $1+\epsilon$ speed online algorithms [BCKN05].

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 [EP09]. Our main idea is to bypass the [EP03] reduction (from broadcast scheduling to non-clairvoyant scheduling) that loses a factor of 2 in the speedup and directly adapt those ideas to broadcast scheduling. The algorithm and its analysis are actually very simple. Our approach is the following: We first consider the *fractional* version of the problem (i.e. pages can be fractionally transmitted in each time-slot) and show that a variant of LAPS (adapted to the broadcast setting) is $(1+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive. Note that this guarantee matches that for LAPS. 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 [EP03, EP05], but one main difference in our work is that we consider a different fractional relaxation, and this enables us to obtain a fully scalable algorithm.

Our algorithm and its analysis can be extended to a more general setting where the pages have arbitrary sizes, and the requests have dependencies. As mentioned earlier (and we describe an example in Section 5.3 for completeness), in order to obtain any reasonable guarantees under arbitrary page-sizes, one needs to consider the preemptive version, i.e. transmissions of a need not occur consecutively. We emphasize that in a valid preemptive schedule only one page is transmitted in each time-slot; however since pages have arbitrary sizes, complete transmission of a page may involve several (non-consecutive) time-slots. Hence a preemptive schedule differs from a 'fractional schedule', where several pages may be fractionally transmitted in the same time-slot. Notice that 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 [IM10], 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 $O(k \log n)$.

Remark: Although we state the above result only for the (more restrictive) version where there is no cache available, the approximation guarantee in Theorem 1.3 holds relative to an optimal schedule for the (less restrictive) version where cache is available. Thus Theorem 1.3 implies scalable online algorithms for both versions, with and without caching. For the setting where cache is available, the problem can even be reduced to dependent requests with single sized pages, since we can replace a page p of length ℓ_p by ℓ_p unit size pages, and modify any original request for p to now request the corresponding ℓ_p unit size pages.

Given the results above, a natural question is whether the loss of factor 2 speed up in previous approaches [EP03, EP05] can be avoided in the reduction from broadcast scheduling to the non-clairvoyant scheduling problem. 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 speed up or in the competitive ratio (i.e. it is a $(1,1)$ transformation). Again, the main idea to achieve this lies in the appropriate definition of the fractional broadcast problem, and the online rounding algorithms required to reduce the broadcast problem to its fractional relaxation. Note that this reduction combined with LAPS [EP09] would also imply our results. However, in this paper we have chosen to present our results directly without going via the non-clairvoyant reduction, since the proofs seem much simpler and cleaner with this approach. We note that this reduction could be useful in other contexts, and present it for completeness in Section 6.

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.

2 Fractional Broadcast Scheduling

In this section we study a "continuous" variant of the broadcast scheduling problem and obtain a $(1+\epsilon)$-speed, $O(1/\epsilon^2)$-competitive algorithm for it. In the next two sections, we will show how to transform this algorithm into one for the actual broadcast problem. To obtain the randomized algorithm in Section 3, we use an α-point randomized rounding technique from [BCKN05]. To obtain the deterministic algorithm in Section 4, we present a different technique that loses an additional factor of $1/\epsilon$.

The setting for the *fractional* broadcast scheduling problem is the same as that for usual broadcast scheduling, namely a single server has n pages and requests for pages arrive online. The key point is that the fractional schedule can transmit pages in a continuous manner. 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 $\sum_{p=1}^{n} x_p(t) \leq 1$. Again in the resource augmentation setting, a $(1+\epsilon)$-speed schedule has $\sum_p x_p(t) \leq 1+\epsilon$ 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, $\frac{1}{m}\sum_{r\in[m]}(b(r)-a(r))$.

2.1 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 fractionally completed. Let $N'(t)$ denote the $\epsilon|N(t)|$ "most-recent" requests among $N(t)$, i.e. those with the latest arrival times, with ties broken arbitrarily. The algorithm then time shares among $N'(t)$, i.e.

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

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

2.2 Analysis for Fractional Broadcast

Our analysis is based on a potential function argument inspired by that for LAPS [EP09]. 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 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.

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

Here, $\Delta\mathsf{On}(t)$ and $\Delta\mathsf{Opt}(t)$ denote the additional cost incurred in an infinitesimal time interval $[t, t+dt)$ by the online schedule and the optimal schedule respectively, and $\Delta\Phi(t)$ is the change in potential function. Moreover, we will ensure that $\Phi(0) = \Phi(\infty) = 0$. It can then be easily seen that this implies 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 $\mathsf{Opt}(p, t_1, t_2) := \int_{t_1}^{t_2} x_p^*(t)dt$ denote the *amount of page* p transmitted by Opt in the interval $[t_1, t_2]$.

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 $\mathsf{On}(r, t_1, t_2) := \int_{t_1}^{t_2} y_r(t)dt$, i.e. the fractional time that the online algorithm has devoted towards *request* r in the interval $[t_1, t_2]$. Observe that any request r is inactive after time $b(r)$, and hence $y_r(t) = 0$ for all $t > b(r)$. Thus $\mathsf{On}(r, t, \infty) = \mathsf{On}(r, t, b(r))$ for all $r \in [m]$ and $t \geq 0$. At any continuous time t, let $N(t)$ and $N^*(t)$ respectively denote the set of requests that are not yet completed in On and Opt.

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

$$z_r(t) = \mathsf{On}(r, t, \infty) \cdot \mathsf{Opt}(p(r), a(r), t)$$

Note that $z_r(t) \geq 0$ for any r and 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 request which arrived most recently and a rank of 1 to the oldest active request).

We now show that (2.1) holds.

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 Algorithm 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 $z_{r'}(t) \geq 0$ for any r', the contribution due to these requests to the potential can only decrease. Thus $\Delta\Phi \leq 0$. And again, at that instant, $\Delta\mathsf{On} = \Delta\mathsf{Opt} = 0$, and hence equation (2.1) 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 Opt *broadcast in* $(t, t+dt)$*:* We will show that $\Delta\Phi(t) \leq \frac{1}{\epsilon}|N(t)|dt$. To see this, consider any page p. The amount of page p transmitted by Opt in this interval is $x_p^*(t)dt$. This broadcast of $x_p^*(t)dt$ amount 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. That is, the page p transmission increases potential corresponding to the requests:

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

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

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

We now show that $\sum_{r \in C(t,p)} \mathsf{On}(r,t,\infty) \le 1$ for any page p. Let $r' = \arg\max\{b(r) \mid r \in C(t,p)\}$ be the request in $C(t,p)$ which is completed last by On. Since r' is *not* completed until $b(r')$ and $a(r') \le t$, it must be that On broadcasts at most 1 unit of *page* p during $[t, b(r')]$; otherwise $b(r')$ would be smaller. Hence $\sum_{r \in C(t,p)} \mathsf{On}(r,t,b(r')) \le 1$. Observe that for all $r \in C(p,t)$ and $t \ge b(r')$, we have $y_r(t) = 0$ since $b(r) \le b(r')$. Thus $\sum_{r \in C(t,p)} \mathsf{On}(r,t,\infty) \le 1$. Now plugging this into equation (2.2), we have that

$$\Delta\Phi \le \frac{1}{\epsilon}|N(t)| \cdot \sum_{p=1}^{n} x_p^*(t) dt \le \frac{1}{\epsilon}|N(t)| \cdot dt \tag{2.3}$$

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 $y_r(t)$ 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 $\mathsf{Opt}(p(r), a(r), t) \ge 1$ since Opt has completed request r by time t. Thus $z_r(t) \ge \mathsf{On}(r,t,\infty)$. Note also that $y_r(t) = (1+4\epsilon)/|N'(t)|$. Thus,

$$\frac{d}{dt} z_r(t) \le \frac{d}{dt}\mathsf{On}(r,t,\infty) = -y_r(t) = -\frac{1+4\epsilon}{|N'(t)|}, \qquad \text{for all } r \in N'(t) \setminus N^*(t).$$

Furthermore, since each request that On works on in $[t, t+dt)$ has rank at least $(1-\epsilon) \cdot |N(t)|$, the potential Φ increases at rate,

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

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

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

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 we obtain Theorem 2.1.

Note the the resulting fractional broadcast schedule may complete requests at fractional times. However, even if we round up these completion times to integer values, the average flow-time objective only increases by a constant factor. 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.

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 2.1. Our rounding technique requires a speed up of $(1 + \epsilon)$, and loses a factor of $O(1/\epsilon)$ (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). Our technique is similar to that used by Edmonds and Pruhs [EP03] to convert their "fractional" algorithm B-EQUI to B-EQUI-EDF. However, there are some crucial differences, in particular since our notion of fractional schedule is somewhat different (it is precisely the difference that allows us to avoid the loss of factor 2 in the speed up required).

3.1 Algorithm

Let On denote the fractional algorithm Recalling the notation from the previous section, $a(r)$ 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 request that are as yet unsatisfied requests by Rnd by have been fractionally satisfied by On. At any time, it transmits the request from the queue with the least width. Formally, at time t, the algorithm operates as follows:

Algorithm 1 OnlineRounding(t)

1: **for** any request r that completes under 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 $\lceil \frac{1}{\epsilon} \rceil$.

3.2 Analysis

We will show that

Theorem 3.1 *Given any fractional broadcast schedule, the above algorithm produces an integral schedule such that*

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

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

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 $\left[a(r), b(r) + \frac{2}{\epsilon}(b(r) - a(r)) + 2\right]$.

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.

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

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 $[t_\ell, t]$, the algorithm always dequeues requests of width at most $w(r)$. We will show that there exists time $t' \leq t_\ell + \frac{2w(r)}{\epsilon} + 2$, 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 $[b(r), t_\ell + \frac{2w(r)}{\epsilon} + 2]$, 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 $\mathcal{T} := [t_\ell, t_\ell + \frac{2w(r)}{\epsilon} + 2]$. 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, $\int_{t_1}^{t_2} x_p(t)dt \geq 1$.*

Proof. Since 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 $\int_{t_\ell - w(r)}^{t_p} x_p(t)dt \geq 1$.*

Proof. By our assumption $\mathcal{T}$, 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, $b(r') \in [t_\ell, t_p]$: indeed if $b(r') < t_\ell$, then the queue would have contained a request of width at most $w(r)$ at time $t_\ell - 1$, contradicting the definition of t_ℓ. This implies that $a(r') \geq b(r') - w(r) \geq t_\ell - w(r)$. Thus

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

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 $t_\ell - w(r)$ 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 $[t_\ell - w(r), t_\ell] \cup \mathcal{T}$. Thus $\sum_p N_p \leq |\mathcal{T}| + w(r)$. On the other hand, as Rnd runs at speed $(1+\epsilon)$ and $\mathcal{Q}$ is never empty during $\mathcal{T}$, which implies that $\sum_p N_p \geq (1+\epsilon)|\mathcal{T}| - 1$. These two bounds imply that $|\mathcal{T}| + w(r) \geq (1+\epsilon)|\mathcal{T}| - 1$ which implies that $|\mathcal{T}| \leq (w(r)+1)/\epsilon \leq 2w(r)/\epsilon$, contradiction our assumption that $\mathcal{T}$ has length $2w(r)/\epsilon + 2$. ■

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.

The rounding algorithm is based on the α-point rounding technique. This result is originally from Bansal et al. [BCKN05]; however we present it here for completeness since the proof as presented in [BCKN05] appears incorrect; it claims an additive guarantee of $O(1/\epsilon)$, though the proof only seems to imply a guarantee of $O(1/\epsilon^2)$.

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, otherwise we can rescale the units of speed. For page $p \in [n]$ and times $t_1 < t_2$, let $\mathsf{On}(p, t_1, t_2) = \int_{t_1}^{t_2} x_p(t)dt$ denote the (fractional) amount of page p broadcast in the interval $[t_1, t_2)$. The algorithm works as follows:

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 $\mathsf{On}(r(p), a(r), b(r)) = 1$. Thus there exists some (fractional) time $t \in (a(r), b(r))$ such that $\mathsf{On}(r(p), 0, t) \in \alpha_p + \mathbb{Z}_+$. Since $a(r)$ and $b(r)$ are integral, the claim follows. ■

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 α_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 $\{p \in [n] \mid \exists i \in \mathbb{Z}_+,\ \mathsf{On}(p, 0, t-1) < i + \alpha_p \leq \mathsf{On}(p, 0, t)\}$.
    dequeue the first page in $\mathcal{Q}$, and broadcast it. If t is a multiple of $\lceil \frac{1}{\delta} \rceil$ perform this step twice.
end for

Next we bound the expected time spent by each page in the queue. First, the following lemma from [BCKN05] shows that it suffices to consider the expected queue length at any time t.

Lemma 4.2 *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 queue length at any time $\mathcal{Q}$.

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 [BCKN05]. $\mathcal{Q}_t$ 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. 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 $a_p = \mathsf{On}(p, t-k, t)$, and random variable A_p denote the number of enqueues of page p during $[t-k, t]$. Since the αs for different pages are chosen independently, A_ps are independent rvs. Additionally, by α-point rounding we have $A_p \in \{\lfloor a_p \rfloor, \lceil a_p \rceil\}$ for all $p \in [n]$; and $E[\sum_{p=1}^n A_p] = \sum_{p=1}^n a_p$. Thus 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 [AS00], $Pr[H_0] \leq \exp(-k/2)$, 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 $a_p := \mathsf{On}(p,t-jk-k,t)$, 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:

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

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

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

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

We first define the non-uniform broadcast scheduling problem with dependencies: There are n pages with each page p having an integer size l_p; page p consists of l_p distinct units/positions that are numbered 1 to l_p. Requests for *subsets* of these pages arrive over time. There is a single server that can broadcast pages: in each time-slot the server broadcasts some position of some page. Any request r is specified by its arrival time $a(r)$ and the set of pages $\mathcal{P}(r) \subseteq [n]$ that it requests; we let $[m]$ denote the set of all requests. A broadcast schedule is an assignment of page-positions (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 positions of page p (not necessarily in consecutive time-slots, but in the order 1 through l_p). This is a preemptive schedule since we allow non-contiguous transmission of pages. 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 with non-uniform page sizes, 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.

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 fractional broadcast (unlike the original broadcast problem), we do not have the notion of l_p distinct positions of each page p; we only require the schedule to broadcast l_p indistinguishable units for page p.

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

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

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 $N'(t)$.

5.2 Analysis of Fractional Broadcast

The analysis is very similar to that for the uniform broadcast scheduling case presented in Section 2.2. We first describe the potential function, and then use it to bound the competitive ratio.

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 $\mathsf{On}(r, p, t_1, t_2) := \int_{t_1}^{t_2} y_{r,p}(t)dt$, i.e. the fractional time that the online algorithm has devoted towards *page p on behalf of request r* in the interval $[t_1, t_2]$. Observe that since any request r is inactive after time $b(r)$, we have $y_{r,p}(t) = 0$ for all $t > b(r)$ and $p \in \mathcal{P}(r)$. Thus $\mathsf{On}(r, p, t, \infty) = \mathsf{On}(r, p, t, b(r))$ for all $r \in [m]$, $p \in \mathcal{P}(r)$, and $t \geq 0$.

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 $\sum_p x_p^*(t) \leq 1$ since the offline optimal is 1-speed. For page $p \in [n]$ and times $t_1 < t_2$, let $\mathsf{Opt}(p, t_1, t_2) := \int_{t_1}^{t_2} x_p^*(t)dt$ 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.

$$z_{r,p}(t) = \frac{\mathsf{On}(r, p, t, \infty) \cdot \mathsf{Opt}(p, a(r), t)}{l_p}$$

The total contribution of request r is then $z_r(t) = \sum_{p \in \mathcal{P}(r)} z_{r,p}(t)$. Note that $z_r(t) \geq 0$ for any r and 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 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 2.2, 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.

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

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 $z_{r'}(t) \geq 0$ for any r', the contribution due to these requests to the potential can only decrease. Thus $\Delta\Phi \leq 0$. 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 $\Delta\Phi(t) \leq \frac{1}{\epsilon}|N(t)|dt$. To see this, consider any page p. The amount of page p transmitted by Opt in this interval is $x_p^*(t)dt$. This broadcast of $x_p^*(t)dt$ amount 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,

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

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

$$\Delta\Phi \leq \frac{1}{\epsilon}|N(t)| \cdot \sum_p \sum_{r \in C(t,p)} \frac{\mathsf{On}(r,t,\infty)x_p^*(t)dt}{l_p}. \tag{5.7}$$

We show that $\sum_{r \in C(t,p)} \mathsf{On}(r,t,\infty) \leq l_p$ 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 $a(r') \leq t$, 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 $\sum_{r \in C(t,p)} \mathsf{On}(r,t,b(r',p)) \leq l_p$. Observe that for all $r \in C(p,t)$ and $t \geq b(r',p)$, we have $y_{r,p}(t) = 0$ since $b(r,p) \leq b(r',p)$. Thus $\sum_{r \in C(t,p)} \mathsf{On}(r,t,\infty) \leq l_p$. Now plugging this into equation (5.7), we have that

$$\Delta\Phi \leq \frac{1}{\epsilon}|N(t)| \cdot \sum_p x_p^*(t)dt \leq \frac{1}{\epsilon}|N(t)| \cdot dt \tag{5.8}$$

Recall that $\sum_p x_p^*(t) \leq 1$ 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 $p \in \mathcal{P}(r)$, $\mathsf{Opt}(r,p,t) \geq l_p$ since Opt has completed request r. Thus $z_r(t) \geq \sum_{p \in \mathcal{P}(r)} \mathsf{On}(r,p,t,\infty)$. Note also that $\sum_{p \in \mathcal{P}(r)} y_{r,p}(t) = (1+4\epsilon)/|N'(t)|$. Thus,

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

Furthermore, since each request that On works on in $[t, t+dt)$ has rank at least $(1-\epsilon) \cdot |N(t)|$, the potential Φ increases at rate,

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

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

$$\frac{d}{dt}\Phi(t) \leq -\left(\frac{1}{\epsilon}+2\right)|N(t)| + \frac{1}{\epsilon^2}(1+4\epsilon)|N^*(t)| \leq -\left(\frac{1}{\epsilon}+1\right)\cdot|N(t)| + \frac{2}{\epsilon^2}\cdot|N^*(t)|. \quad (5.9)$$

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),

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

which proves Equation (5.6). Thus we obtain:

Theorem 5.1 *For any $0 < \epsilon \leq \frac{1}{4}$, 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 an online deterministic fashion. Formally, 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

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

where $b_I(r)$ (resp. $b(r)$) is the completion time of request r in the integral (resp. fractional) schedule. 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 positions 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 [EP03, RS07], the algorithm here is much more subtle. The algorithm we present below is an extension of that discussed in Section 3, and requires considerably more work.

For any request $r \in [m]$ and page $p \in \mathcal{P}(r)$, *job* $\langle r,p\rangle$ denotes the page p request 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. $b(r,p)$). 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 position of page p to transmit. At each time-slot, the deterministic schedule broadcasts the current position 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 how much fraction of this page has been transmitted since time s.

Algorithm 3 GenRounding(t)

initialize all jobs as unmarked when they arrive.
simulate fractional online algorithm to obtain schedule On.
for any *unmarked* job $\langle r, p\rangle$ that completes under On at time t, i.e. $b(r,p) = t$, **do**
  if there is a tuple $\tau = \langle p, w, s, i\rangle \in \mathcal{Q}$ of page p with $s \geq a(r)$ **then**
    update the width of tuple τ to $\min(w, b(r,p) - a(r))$.
  else
    insert new tuple $\langle p, b(r,p) - a(r), \infty, 1\rangle$ into $\mathcal{Q}$.
  end if
end for
dequeue the tuple $\tau = \langle p, w, s, i\rangle$ that has least width amongst all elements in $\mathcal{Q}$.
broadcast position i of page p in this time-slot.
if broadcast of p corresponding to τ is just beginning (i.e. $i = 1$) **then**
  mark set $s = t$, i.e. equal to the current time slot .
end if
if broadcast of p corresponding to τ is complete (i.e. $i = l_p$) **then**
  mark all jobs $\langle r', p\rangle$ of page p having $a(r') \leq s$.
else
  enqueue the modified tuple $\langle p, w, s, i+1\rangle$ into $\mathcal{Q}$.
end if

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

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

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 $w \leq b(r,p) - a(r)$. Define,

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

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

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 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 $b(r', q) \le a(r') + w \le s(\tau) + w$. Observe that at time t_A, queue $\mathcal{Q}$ contains no tuple of width at most w. Thus $b(r', q) \ge t_A$, i.e. $s(\tau) \ge t_A - w$, which proves the claim. ■

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

Claim 5.3 *For any* $1 \le j \le N_q - 1$*, we have* $\mathsf{On}(q, s(\tau_j), s(\tau_{j+1})) \ge l_q$*.*

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 $s(\tau_j) < t' \le s(\tau_{j+1})$; 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 $\mathsf{On}(q, s(\tau_j), s(\tau_{j+1})) \ge \mathsf{On}(q, a(r'), b(r', q)) \ge l_q$. ■

Claim 5.4 $\mathsf{On}(q, t_A - w, t_C) \ge l_q$*, 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 $b(r', q) \ge t_A$ and so $a(r') \ge t_A - w$. We consider two cases:

1. $s(\tau_1) \le t_A$. In this case, we have $a(r') \le s(\tau_1) \le t_A$ and so $b(r', q) \le a(r') + w \le t_A + w$. Thus $\mathsf{On}(q, t_A - w, t_A + w) \ge \mathsf{On}(q, a(r'), b(r', q)) \ge l_q$.
2. $s(\tau_1) > t_A$. Since start-time of tuple τ_1 lies in $\mathcal{T}$, and for job $\langle r', q\rangle$ we have $b(r', q) \le s(\tau_1)$. Thus in this case, $\mathsf{On}(q, t_A - w, s(\tau_1)) \ge \mathsf{On}(q, a(r'), b(r', q)) \ge l_q$.

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:

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

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

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

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 $O(k \log n)$. Note 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 number of preemptions decreases dramatically and the current analysis carries through with some minor modifications. We will include a full discussion and proof of this in a later version of this paper.

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, there for any arbitrarily large parameters b and c, there is an adversarial instance such that the online algorithm is at least b competitive even if it has a speed up of factor c.

Consider the following adversarial input: At time $t=0$, there is one request for page p_0 of size $2c$. Then, at each time slot i for $i = 1, 2, 3, \ldots$, there are cbi requests for page p_i, where page p_i has size $\frac{1}{c^2 b^2 i^3}$. The adversary stops giving any requests at a time t_f when *(i)* the online algorithm schedules the page p_0, or *(ii)* the time $t_f = b^2 c^5$.

The argument is based on two cases, depending on how soon the online algorithm schedules p_0. If it is the former case, then the broadcast of the page p_0 must have spanned over two consecutive time slots (even with c speed). Therefore, the cost of the online algorithm is at least cbt_f (all the jobs released at the beginning of time slot $[t_f, t_f+1)$ wait for at least 1 complete time slot). On the other hand, the adversary could schedule the broadcast of a small page p_i as soon as its requests arrive at time i, and then schedule the page p_0 from time t_f+1. The cost it incurs would be at most

$$\left(\sum_{i=1}^{t_f} cbi \cdot \frac{1}{cbi^3}\right) + (t_f + 2c) \leq 2t_f + 2c.$$

Therefore the cost of the online algorithm is at least b times the cost of the optimal solution even if the online algorithm has a speed-up of c.

In the other case, if the online algorithm has not broadcast p_0 until time $t_f = b^2 c^5$, then its flow time is at least $b^2 c^5$. On the other hand we claim that offline incurs a cost of at most bc^4. Consider the solution that broadcasts p_0 in the first $2c$ timeslots, and then the other pages. Since the sum of their sizes $\sum_{i=1}^{2c+1}(1/c^2b^2i^3) \leq 1$, it follows that all the pages $p_1, p_2, \ldots, p_{2c}$ and p_{2c+1} can be broadcast in the next time slot after completing p_0. Therefore, the requests corresponding to $p_1, p_2, \ldots, p_{2c}$ incur a waiting time of at most $(2c+1)$, and all subsequent requests (for pages $p_{2c+1}, p_{2c+2}, \ldots$) incur a collective waiting time of at most $\sum_{i=2c+1}^{t_f} cbi \cdot (1/c^2b^2i^3) \leq 1$, since these pages can be broadcast immediately after the request arrives. The cost of this optimal solution is therefore at most $2c + cb(1+2+\ldots+2c)\cdot(2c+1)+1 \leq 9c^4b$, which implies the claim.

6 Broadcast Scheduling to Non-Clairvoyant Unicast Scheduling

The non-clairvoyant unicast model (stated in a more general form in [EP03]) 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 at for any phase of a job). 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 $\beta \in [0,1]$. 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 phase it is in or how long each phase is, etc.

In [EP03], Edmonds and Pruhs 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 serial 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 [EP03], except for modifications that utilize our definition of fractional broadcast (that differs from [EP03]). 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. We now 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.

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 $j(q)$ is *not* yet completed in $\mathcal{I}'$.
  end for
  for each job $j(r)$ in $\mathcal{I}'$ with $y_{j(r)}(t) > 0$ and $C(r) < t$ **do**
    inform $\mathcal{A}$ that job $j(r)$ is completed.
  end for
end for

We then show that the following inequalities hold.

$$\mathsf{Opt}(\mathcal{I}') \leq \mathsf{Opt}(\mathcal{I}) \tag{6.12}$$

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

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 c-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 $C(r) < C^*(r)$ then $j(r)$ has only a serial phase of duration $C(r) - a(r)$.
- *Type 2 jobs.* If $C(r) \geq C^*(r)$ then $j(r)$ has a serial phase of duration $C^*(r) - a(r) - 1$ followed by a parallel phase with work $\int_{C^*(r)-1}^{C(r)} y_{j(r)}(t)dt + \delta$. Here $\delta > 0$ is infinitesimally small.

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 in $\mathcal{B}$ at time $C(r)$, we declare completion of the corresponding job $j(r)$ in $\mathcal{A}$ only at the earliest time after $C(r)$ when $\mathcal{A}$ schedules $j(r)$ to some infinitesimally small extent δ. 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 the flow time incurred by* r *in* $\mathsf{Opt}(\mathcal{I})$*. Thus* $\mathsf{Opt}(\mathcal{I}') \leq \mathsf{Opt}(\mathcal{I})$*.*

Proof. Firstly observe that in any schedule for $\mathcal{I}'$, the flow time for any type 1 job $j(r)$ equals $C(r) - a(r) < C^*(r) - a(r)$, i.e. it is at most the flow time of request r under $\mathsf{Opt}(\mathcal{I})$. Thus it suffices to bound the flow-time for type 2 jobs.

We create schedule $\mathsf{Sch}(\mathcal{I}')$ for $\mathcal{I}'$ that at any 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 that were satisfied by this broadcast of page p. For each $r \in C(t, p)$ where job $j(r)$ is of type 2, schedule $\int_{C^*(r)-1}^{C(r)} y_{j(r)}(\ell) d\ell + \delta$ units of $j(r)$, i.e. all the parallel work of $j(r)$.

We now show that $\mathsf{Sch}(\mathcal{I}')$ is a feasible $(1 + n\,\delta)$-speed schedule. Taking δ to be infinitesimal, we would obtain a 1-speed schedule. To show feasibility, consider the total work packed in any integral time interval $(t-1, t]$. From the above it is at most $\sum_{r \in C(p,t)} \int_{C^*(r)-1}^{C(r)} y_{j(r)}(\ell) d\ell + n\,\delta$.

For any $r \in C(t, p)$, let $W_r := \int_{C^*(r)-1}^{C(r)} y_{j(r)}(\ell) d\ell = \int_{t-1}^{C(r)} y_{j(r)}(\ell) d\ell$. We claim that $\sum_{r \in C(t,p)} W_r \leq 1$, which implies the feasibility of $\mathsf{Sch}(\mathcal{I}')$. 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 $[t-1, C(r'))$. Hence the total (parallel) work done by $\mathcal{A}$ on jobs corresponding to $C(t, p)$ is $\sum_{r \in C(t,p)} \int_{t-1}^{C(r')} y_{j(r)}(\ell) d\ell \leq 1$. For all $r \in C(t, p)$, since $C(r) \leq C(r')$ we obtain $W_r \leq \int_{t-1}^{C(r')} y_{j(r)}(\ell) d\ell$, which implies the desired claim.

Additionally, note that $\mathsf{Sch}(\mathcal{I}')$ performs parallel work on any type 2 job $j(r)$ only after time $C^*(r) - 1$, i.e. after the serial phase of $j(r)$. Thus $\mathsf{Sch}(\mathcal{I}')$ is indeed feasible.

Next, we argue that the flow time for each type 2 job $j(r)$ in $\mathsf{Sch}(\mathcal{I}')$ is at most the flow time for r in $\mathsf{Opt}(\mathcal{I})$. Let $t = C^*(r)$. By the above definition, the entire parallel work of $j(r)$ is completed during $(t-1, t]$. Thus its flow time in $\mathsf{Sch}(\mathcal{I}')$ is at most $t - a(r) = C^*(r) - a(r)$ which equals the flowtime of r under $\mathsf{Opt}(\mathcal{I})$. Hence we obtain $\mathsf{Opt}(\mathcal{I}') \leq \mathsf{Sch}(\mathcal{I}') \leq \mathsf{Opt}(\mathcal{I})$. ■

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* c*-competitive (w.r.t. 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 [EP09], 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 [RS07] 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 *Any $o(m^{1/3})$-approximation algorithm for disjunctive broadcast scheduling with ρ speed-up implies a 4ρ-approximation algorithm for set-cover (here m is 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

[AS00] N. Alon, and J. Spencer, The Probabilistic Method. *John Wiley & Sons*, 2000.

[BCKN05] N. Bansal, M. Charikar, S. Khanna and J. Naor. Approximating the average response time in broadcast scheduling. In *SODA 05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 215221, 2005.

[BCS06] N. Bansal, D. Coppersmith and M. Sviridenko. Improved approximation algorithms for broadcast scheduling. In *SODA 06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithms*, pages 344353, 2006.

[BM00] Y. Bartal and S. Muthukrishnan. Minimizing maximum response time in scheduling broadcasts. *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[CEHK08] J. Chang, T. Erlebach, R. Gailis, and S. Khuller. Broadcast scheduling: algorithms and complexity. In *SODA 08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 473482, 2008.

[CK06] M. Charikar and S. Khuller. A robust maximum completion time measure for scheduling. *In SODA 06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithms*, pages 324333, 2006.

[CIM09a] C. Chekuri, S. Im and B. Moseley. Minimizing Maximum Response Time and Delay Factor in Broadcast Scheduling. *European Symposium on Algorithms*, 2009.

[IM10] S. Im and B. Moseley. An Online Scalable Algorithm for Average Flowtime in Broadcast Scheduling. *In SODA 10: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, 2010.

[CIM09b] C. Chekuri, S. Im and B. Moseley. Longest Wait First and Broadcast Scheduling. *Workshop on Approximation and Online Algorithms*, 2009.

[CM09] C. Chekuri and B. Moseley. Online scheduling to minimize the maximum delay factor. *In SODA 09: Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, 2009.

[E00] J. Edmonds. Scheduling in the dark. *Theor. Comput. Sci.*, 235(1):109141, 2000.

[EP03] J. Edmonds and K. Pruhs. Multicast pull scheduling: When fairness is fine. *Algorithmica*, 36(3):315330, 2003.

[EP05] J. Edmonds and K. Pruhs. A maiden analysis of longest wait first. *ACM Trans. Algorithms*, 1(1):1432, 2005.

[EP09] J. Edmonds and K. Pruhs. Scalably scheduling processes with arbitrary speedup curves. *In SODA 09: Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, 2009.

[EH02] T. Erlebach and A. Hall. NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *Proc. 13th ACM-SIAM Symp. on Disc. Algorithms*, 2002, pp. 194-202.

[GKKW04] R. Gandhi, S. Khuller, Y. Kim, and Y. (Justin) Wan. Algorithms for minimizing response time in broadcast scheduling. *Algorithmica*, 38(4):597608, 2004.

[GKPS06] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324360, 2006.

[KP00] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617643, 2000.

[KPV00] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai, Scheduling broadcasts in wireless networks, *Proc. of the 8th Annual European Symposium on Algorithms*, 2000, pp. 290-301.

[KC04] J. Kim and K. Chwa. Scheduling broadcasts with deadlines. *Theor. Comput. Sci.*, 325(3):479488, 2004.

[PST04] K. Pruhs, J. Sgall and E. Torng. Online Scheduling. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, editor Joseph Y-T. Leung, CRC Press, 2004.

[PU05] K. Pruhs and P. Uthaisombut. A comparison of multicast pull models. *Algorithmica*, 42(3-4):289307, 2005.

[RS07] N. Schabanel and J. Robert. Pull-Based Data Broadcast with Dependencies: Be Fair to Users, not to Items. In *Proc. of the 18th ACM/SIAM Symp. on Discrete Algorithms, SODA 2007*, pages 238–247.

[ZFCCPW06] F. Zheng, S.P.Y. Fung, W.T. Chan, F.Y.L. Chin, C.K. Poon and P. Wong. Improved On-line Broadcast Scheduling with Deadlines. *Proceedings of the 11th Annual International Computing and Combinatorics Conference (COCOON)*, 2006, pp. 320–329.