Operations Research Letters 44 (2016) 784–789

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Minimizing the maximum flow time in batch scheduling

Sungjin Im [a,*], Hoon Oh [b], Maryam Shadloo [a]

[a] University of California at Merced, Merced, CA, United States
[b] Rutgers University-Camden, Camden, NJ, United States

ARTICLE INFO

Article history:
Received 27 May 2016
Received in revised form
29 September 2016
Accepted 29 September 2016
Available online 6 October 2016

Keywords:
Batch scheduling
Broadcast scheduling
Maximum flow time
Approximation
Resource augmentation

ABSTRACT

We consider the maximum flow time minimization problem in batch scheduling, which is a capacitated version of broadcast scheduling. In this setting, n different pages of information are available at the server which receives requests from clients over time for specific pages. The server can transmit at most one page p at each time to satisfy a batch of requests for the same page p, up to a certain capacity B_p. In this paper we give the first $(1+\epsilon)$-approximations for this problem with arbitrarily small resource augmentation, using either more capacity or more speed.

1. Introduction

In batch scheduling, there is a server that stores n different unit-sized pages of information. Each client submits to the server a request ρ at time r_ρ asking for a specific page p_ρ. The server can transmit at most one page p at each time to satisfy up to B_p outstanding requests of the same page p simultaneously; the capacity B_p can be different for each page. Processing a batch of requests together is a popular method to increase the server's throughput. Not surprisingly, batch scheduling appears in various forms in numerous applications, not only in server–client scheduling, but also in manufacturing lines; for pointers of the applications, see [7].

Broadcast scheduling is a special case of batch scheduling which has received considerable attention in theoretical computer science. The only difference is that in broadcast scheduling there is no limit on the number of requests the server can aggregate at a time, i.e. $B_p = \infty$ for all p. In other words, batch scheduling is a capacitated version of broadcast scheduling. However, as discussed in [2], capacities are often present in practice. For example, there could be a limit on the number of clients a server can serve at a time.

What makes batch/broadcast scheduling algorithmically challenging is that the scheduler could aggregate more requests by waiting for other requests arriving in the future for the same page. While the server can increase throughput by doing so, it makes earlier arriving requests wait longer, thereby making the clients submitting those requests unhappy. Such a tradeoff becomes more challenging in batch scheduling since the scheduler also has to factor in batch sizes. A request ρ's flow time is defined as its completion time C_ρ minus its arrival time r_ρ and measures how long the request waits since its arrival until its completion time. When requests compete to get served earlier, a popular way of combining the flow time of individual requests is to consider flow time objectives such as total flow time or the maximum flow time.

In this paper, we study the objective of minimizing the maximum flow time, i.e. $\max_\rho(C_\rho - r_\rho)$ in the batch scheduling setting. In broadcast scheduling, First-In-First-Out (FIFO) is known to be a 2-approximation [3,5]. At each time, the algorithm FIFO transmits the page of an outstanding request with the earliest arrival time; notice that FIFO is in fact an online algorithm since it does not need to know requests arriving in the future. It was subsequently shown that no online algorithms can be better than 2-competitive [3,4]. The open question whether there exists a better than 2-approximation was recently answered in [7], which gave a PTAS using a variant of α-point rounding and dynamic programming (DP). The work in [7] essentially closed the complexity of the problem in broadcast scheduling since the problem was already known to be strongly NP-hard [3].

The main goal of this paper is to understand the complexity of the maximum flow time objective in batch scheduling, which captures capacity constraints commonly appearing in practice. A recent work shows that FIFO is still 2-competitive in batch

* Corresponding author.
E-mail addresses: sim3@ucmerced.edu (S. Im), ho62@rutgers.edu (H. Oh), mshadloo@ucmerced.edu (M. Shadloo).

http://dx.doi.org/10.1016/j.orl.2016.09.016
0167-6377/

scheduling [6] as it is in broadcast scheduling. Our work started from the question if there exists a better than 2-approximation in batch scheduling.

1.1. Our result

Our main result is the first $(1+\epsilon)$-approximations with arbitrarily small resource augmentation. We consider two types of resources augmented, capacity and speed. In the capacity augmentation model, the algorithm is allowed to satisfy up to $(1+\delta)B_p$ requests of page p by one transmission of page p and is compared against the optimal scheduler subject to the original capacity B_p for every p. In this model, if the algorithm's objective is at most c times the optimum for all inputs, we say that the algorithm is a $(1+\delta)$-capacity c-approximation. We believe that capacity augmentation model is reasonable since capacities are specified only approximately in practice when capacities are large—if all capacities are constants, we obtain a PTAS without any resource augmentation; see Section 2.2.

In the speed augmentation model, both the algorithm and the optimal scheduler are subject to the same capacities, but the algorithm is given an extra speed. If it is given $1+\delta$ speed, it is allowed to make one additional transmission than the optimal scheduler in every $\lfloor 1/\delta \rfloor$ time steps. In this model, we say the algorithm is a $(1+\delta)$-speed c-approximation if the algorithm's objective is at most c times the optimum for all inputs. Speed augmentation is widely considered in the scheduling literature [8].

Theorem 1. *Let m denote the number of requests. For minimizing the maximum flow time in batch scheduling, for any $\epsilon > 0$ and $\delta > 0$, we have the following approximations:*

1. *[Section 2.3] a $(1+\delta)$-capacity $(1+\epsilon)$-approximation with running time $m^{O\left(\frac{1}{\epsilon^4\delta^2}\right)}$; and*
2. *[Section 2.4] a $(1+\delta)$-speed $(1+\epsilon)$-approximation with running time $m^{O\left(\frac{1}{\epsilon^3\delta}\cdot\log(1/(\epsilon\delta))\right)}$.*

We also show how to obtain a quasi-polynomial time approximation scheme (QPTAS) without using any extra resources in Section 2.1. Currently, we do not know how to obtain a true PTAS, which we leave as an open problem.

1.2. Overview of our approach

At a high level, we closely follow the PTAS framework used in [7] for broadcast scheduling, which combines DP and a variant of α-point rounding. We discuss how we modify each part to obtain our result in batch scheduling. We first discuss the rounding part. The rounding part is used when the optimum, opt is large, say $\text{opt} \geq \Omega(\log m)$ where m is the number of requests. A standard linear programming (LP) used in broadcast scheduling is the following: variable $x_{p,t}$ denotes how much page p is transmitted at time t, and we need constraints that (i) page p must be transmitted within opt time steps after every time the page is requested, and (ii) at most one page can be transmitted at each time. In batch scheduling, we need to add more constraints to factor in capacity constraints. For every page p and every interval $I = [t_1, t_2]$, we ensure that (i′) at least $\lceil m_{p,I}/B_p \rceil$ transmissions are made for page p during $[t_1, t_2 + \text{opt}]$, where $m_{p,I}$ is the number of requests made during I for page p. This new constraint, together with (ii), turns out to be necessary and sufficient conditions for a feasible integral solution to correspond to a schedule with the maximum flow time at most opt.

We use the same variant of the α-point rounding used in [7]. We give a quick overview of the rounding scheme explaining how it works well with the new LP constraint. After solving the LP, we obtain a fractional solution $\{x_{p,t}\}$ and would like to round it. A standard α-point rounding picks a random value α_p from [0, 1] uniformly and independently for each page p, then attempts to transmit page p at the first time t when the accumulative transmission of page p, i.e. $\sum_{t' \leq t} x_{t'p}$ becomes greater than α_p plus each non-negative integer. Note that a new transmission of page p is made before the LP solution accumulates another unit of transmissions of page p. Hence due to the constraint (i), we obtain a temporary schedule with the maximum flow time at most opt. However, the temporary schedule may be infeasible since it may make too many transmissions during a short interval, which translates into a large increase of the objective when it is converted into a feasible schedule. Roughly speaking, a large number of pages/random variables result in a large variance in congestion. To overcome this issue, [7] partitioned pages into $O(\text{opt})$ groups and used only one random variable of the maximum value at most 1 for each group, therefore was able to have a small congestion over all intervals w.h.p. The new rounding kept the key property that a new transmission of page p is made before the LP solution accumulates another unit of transmissions of page p. Thus, the new constraint (i′) we use for batch scheduling ensures that the rounding scheme makes enough transmissions in the temporary schedule while using a small number of random variables. The rounding can be de-randomized using the method of pessimistic estimators [9].

We now discuss the DP part which is used when $\text{opt} = O(\log m)$. It is easy to see that in an optimal solution, the number of distinct pages of requests alive at a time is at most opt. Using this observation, if $B_p = O(1)$ for all p, one can obtain an optimal schedule via a DP that keeps track of the number of outstanding requests for each page. In the capacity augmentation model, by adding some dummy requests and using an appropriate scaling, we reduce the general problem to the case where all capacities are constants. In the speed augmentation model, we use a different idea. We make an extra transmission of page p before we have too many possibilities for the number of alive requests of the page p—the transmission is used to simplify the number. Here we carefully decide which pages to transmit using extra speed since we are allowed to make only one extra transmission in every $1/\delta$ time steps.

1.3. Related work

In batch scheduling, [6] studied online algorithms for flow time objectives. Specifically, [6] showed $O(1)$-speed $O(1)$-capacity $O(1)$-competitive algorithms for the total flow time objective and the more general ℓ_k-norms of flow time. As mentioned before, [6] also showed that FIFO is 2-competitive for the maximum flow time objective. For the best offline results on flow time objectives in broadcast scheduling, see [1,7].

1.4. Notation and organization

Let $T := \max_\rho r_\rho + m$ be the last time we need to consider in our schedule. In other words, any 'reasonable' algorithm can complete all requests by time T. As observed in [7], one can assume w.l.o.g. that $T = O(m^2)$. This is because if there is an idle time period of length more than m, one can break the instance into two disjoint instances since the earlier arriving requests can be satisfied by any reasonable algorithm before the other requests arrive. We will show algorithms with running time polynomial (or quasi-polynomial) in m and T, which will imply the desired results.

For notational convenience, we will use a model that is slightly different from but equivalent to the previously studied models. At each time, first a set of requests arrive, and then a page is transmitted to satisfy requests that have arrived but have not been

satisfied/completed. Note that in this model a request may have flow time 0. This is not a problem in the study of the maximum flow time objective since we can solve the problem optimally when the optimum is a constant; see Section 2.1. Our algorithm can be easily adapted to the previously studied models.

Consider an arbitrary schedule. We let A_t denote the set of requests alive at time t. When we say that a request ρ is alive at time t, we mean that it is just after we make a transmission at the time. We say that a page p is alive at time t if there is a request in A_t for the page. For a set R of requests, let $P(R)$ denote the pages requested by a request in R. Thus, $P(A_t)$ refers to the set of pages alive at time t. Let $R_{(\cdot)}$ denote the set of requests satisfying the condition specified in the subscript. For example, $R_{r\le t}$ is the set of requests arriving by time t.

Using a standard binary search we can assume w.l.o.g. that we know the value of the optimum, opt. We consider the two cases, $\mathsf{opt} = O(\frac{1}{\epsilon^3}\log T)$ and $\mathsf{opt} = \Omega(\frac{1}{\epsilon^3}\log T)$ in Sections 2 and 3, respectively.

2. Case $\mathsf{opt} = O(1/\epsilon^3)\log T$

In this section we handle the case when $\mathsf{opt} = O(1/\epsilon^3)\log T$ using the (DP). We first study simpler cases when $\mathsf{opt} = O(1)$ or $\max_p B_p = O(1)$ as a warm-up while observing that we can solve the general case optimally in quasi-polynomial time. Then, we proceed to give polynomial time algorithms for the general case in the capacity and resource augmentation models in Sections 2.3 and 2.4, respectively. As we will see in Section 3, the running time of our algorithms will be dominated by that of the DPs we show in this section.

2.1. Warm-up: $\mathsf{opt} = O(1)$

We show that we can solve the problem optimally when $\mathsf{opt} = O(1)$. We say that a set of requests alive at time t, A_t, is achievable if there is a feasible schedule up to time t where all requests in $\mathcal{R}_{r\le t}$, except exactly A_t, are satisfied by time t and have flow time at most opt, and $A_t \subseteq R_{t-\mathsf{opt}<r\le t}$; requests arriving by time $t-\mathsf{opt}$ must be satisfied by time t. Clearly, it must be the case that $|P(A_t)| \le \mathsf{opt}$ since we have to satisfy all requests in A_t within opt time steps. Let $\mathcal{A}_t$ be a collection that includes all possible A_t. We want to bound $|\mathcal{A}_t|$. By expressing A_t as the number of requests for each page in $P(A_t)$, we have that $|\mathcal{A}_t| \le m^{2\mathsf{opt}}$. This is because $|P(A_t)| \le \mathsf{opt}$, each page in $P(A_t)$ has at most m possibilities, and there can be at most m alive requests for each page p.

We construct a DP table with entries $\{\mathrm{DP}(t, A_t)\}_{t, A_t\in\mathcal{A}_t}$ where $A_t \subseteq R_{t-\mathsf{opt}<r\le t}$; note that A_t can only include requests that have been waiting for less than opt times steps since their arrival. From the above discussion, the table size is manageable, at most $m^{2\mathsf{opt}}\cdot T$. Each entry $\mathrm{DP}(t, A_t)$ has value either true or false, and is set to true if and only if A_t is achievable. We can fill out the DP table as follows: $\mathrm{DP}(t, A_t)$ is true if and only if there is an entry $\mathrm{DP}(t-1, A_{t-1})$ with true value such that we can obtain A_t by removing up to B_p requests of a certain page p from $A_{t-1}\cup R_{r=t}$. The time needed to fill out the entire table is only $poly(m)$ factor larger than the DP table size. Notice that there is a schedule with the maximum flow time at most opt if and only if $\mathrm{DP}(T, \emptyset)$ is true—such a schedule, if it exists, can be recovered using a standard traceback method. Therefore, the case that $\mathsf{opt} = O(1)$ can be solved optimally. Note that the table size $T\cdot m^{2\mathsf{opt}}$ is quasi-polynomial in m in the worst case when $\mathsf{opt} = O(\log T)$, thus we can get a QPTAS for the general case, together with the PTAS for the case that $\mathsf{opt} = \Omega(\log T)$ which we will present in Section 3.

2.2. Warm-up: $B := \max_p B_p = O(1)$

In this case we make another observation to reduce the number of possibilities of $P(A_t)$—our argument in the previous section only gives a very loose upper bound of m^{opt}, which is not useful since opt is no longer a constant. Note that $|P(R_{t-\mathsf{opt}<r\le t})| \le 2\mathsf{opt}$ since all requests in $R_{t-\mathsf{opt}<r\le t}$ must be satisfied during $(t-\mathsf{opt}, t+\mathsf{opt}]$. Also we know $P(A_t) \subseteq P(R_{t-\mathsf{opt}<r\le t})$. Thus, the number of possibilities of $P(A_t)$ is upper bounded by $2^{2\mathsf{opt}} = 4^{\mathsf{opt}}$. Now for a fixed $P(A_t)$, we would like to count the number of possible $\{n_{p,t}\}_{p\in P(A_t)}$ where $n_{p,t}$ is the number of requests of page p alive at time t. Knowing that $\sum_{p\in P(A_t)} n_{p,t} \le B\mathsf{opt}$ and $|P(A_t)| \le \mathsf{opt}$, the number of possibilities of $\{n_{p,t}\}_{p\in P(A_t)}$ is at most $\binom{B\mathsf{opt}+\mathsf{opt}}{\mathsf{opt}} \le 2^{(B+1)\mathsf{opt}}$; see Proposition 1. Therefore, we have $|\mathcal{A}_t| \le 2^{(B+3)\mathsf{opt}}$. Since $B = O(1)$ and $\mathsf{opt} = O(\log m)$ for any fixed ϵ, we have a poly-sized DP table which we can complete in polynomial time.

Proposition 1. *Let $z_1, z_2, \ldots, z_k$ be variables which can take non-negative integer values. The number of $\{z_i\}_{i\in[k]}$ satisfying the inequality $\sum_{i=1}^k z_i \le L$ is $\binom{L+k}{k} \le 2^{L+k}$.*

2.3. General case under capacity augmentation

In this section we give a $(1+\epsilon)$-approximation for the case $\mathsf{opt} = O(\frac{1}{\epsilon^3}\log T)$ assuming that our algorithm can handle $(1+\delta)$ larger batches than the optimal scheduler. Our main idea is to cluster requests of the same page to reduce the general case to the constant B case, which we already know how to solve. To streamline our presentation, we assume that $1/\epsilon$, $1/\delta$ and $\epsilon\mathsf{opt}$ are all integers; the last assumption is w.l.o.g. since we can solve the case when $\mathsf{opt} = O(1)$ optimally.

We take the following two preprocessing steps to simplify the instance. We say a time t is a grid time if it is an integer multiple of $\epsilon\mathsf{opt}$.

1. Shift each request ρ's arrival time to the right to the closest grid time, i.e. $\epsilon\mathsf{opt}\left\lceil \frac{r_\rho}{\epsilon\mathsf{opt}}\right\rceil$.
2. Consider each page p with $B_p \ge \frac{2}{\delta\lambda}$ where $\lambda := \delta\epsilon/4$: Let B'_p be the largest integer no greater than B_p that is an integer multiple of $1/\lambda$; note that $B'_p \ge \frac{2}{\delta\lambda}$. Round up $m_{p,t}$, the number of requests arriving for each page p at time t, to the nearest integer multiple of $\lambda B'_p$, i.e. $\lambda B'_p\left\lceil \frac{m_{p,t}}{\lambda B'_p}\right\rceil$. Replace B_p with $\frac{4(1+\delta)}{\epsilon\delta}$ and $m_{p,t}$ with $\left\lceil \frac{m_{p,t}}{\lambda B'_p}\right\rceil$ (scaling down B_p and $m_{p,t}$).

Note that we have $B := \max_p B_p \le \frac{8}{\epsilon\delta^2}$ after the above modifications. Recall that $T = O(m^2)$ and we have shown that the DP is of size at most $T\cdot 2^{(B+3)\mathsf{opt}} = m^{O(\frac{1}{\epsilon^4\delta^2})}$. This shows the running time in the capacity augmentation model claimed in Theorem 1.

It now remains to show that the above reduction is valid. It is obvious that the first modification increases the optimum by at most $\epsilon\mathsf{opt}$ since delaying the entire optimal schedule by $\epsilon\mathsf{opt}$ is a feasible schedule for the modified instance. For notational convenience, assume that there is a schedule with the maximum flow time at most opt for this modified instance; this is w.l.o.g. due to the asymptotic bounds claimed in the theorem.

We now show that even after we increase $m_{p,t}$ for some pages in the second modification, the optimal objective does not increase when strengthened with $(1+\delta)$ capacity augmentation. The schedule will remain the same except that it may have to satisfy more requests for page p by one transmission, namely up to $(1+\delta)B'_p$. To see this, consider a transmission of page p. Say the transmission is made at time t. The transmission satisfies some

requests arriving during $[t-\mathsf{opt}, t]$. Here we let the transmission satisfy the extra requests of page p we added during $[t-\mathsf{opt}, t]$. Hence in the worst case, the number of requests it should satisfy is at most $\lambda B'_p(\frac{1}{\epsilon}+1)+B_p \le \frac{\delta}{2}B'_p+B'_p+1/\lambda \le (1+\delta)B'_p$; there are at most $1/\epsilon+1$ grid times during the interval, requests arrive only at grid times after the first modification, and at most $\lambda B'_p$ requests of page p are added at each grid time. Further, note that since we only increased $m_{p,t}$, a schedule for the modified instance can be easily translated into one for the original instance without increasing the maximum flow time.

Finally, we briefly discuss why we can replace B_p with $\frac{4(1+\delta)}{\epsilon\delta}$ for some pages p. Note that the number of requests of page p arriving at time t, $m_{p,t}$ and $(1+\delta)B'_p$ are both integer multiples of $\lambda B'_p$. What this means is that the number of requests for page p alive at each time is always an integer multiple of $\lambda B'_p$. Hence we can scale down the numbers by a factor of $\lambda B'_p$; notice that $\frac{(1+\delta)B'_p}{\lambda B'_p} = \frac{4(1+\delta)}{\epsilon\delta}$.

2.4. General case under speed augmentation

In this section we give a $(1+\epsilon)$-approximation for the case $\mathsf{opt} = O(\frac{1}{\epsilon^3}\log T)$ assuming that our algorithm can make one additional transmission in every $1/\delta$ time steps. Assume that $\epsilon, \delta \le 1/10$. For simplicity, assume that $1/\delta$, $1/\epsilon$, and $\epsilon\delta\mathsf{opt}$ are all integers. We also assume w.l.o.g. that requests arrive and are satisfied only at grid times (integer multiples of $\epsilon\mathsf{opt}$), and at most $\epsilon\mathsf{opt}$ transmissions are made at each grid time. As observed in [7], this will increase the approximation factor by a factor of at most $(1+2\epsilon)$. We assume that there is a schedule with the maximum flow time opt; this is w.l.o.g. due to the asymptotic bound claimed in Theorem 1. Since we have $(1+\delta)$-speed augmentation, we are allowed to make up to $(1+\delta)\epsilon\mathsf{opt}$ transmissions at each grid time.

In the speed augmentation model, we cannot add a small number of requests at every grid time for each alive page to simplify the instance as we did in the capacity augmentation model. This is because even an extra request may force the algorithm to make one more transmission, but we only have $(1+\delta)$-speed augmentation.

The key idea is to make an extra transmission of page p before the number of possibilities of $n_{p,t}$ becomes unaffordable. Here $n_{p,t}$ denotes the number of requests of page p that are alive at time t in a certain schedule in consideration; this is different from $m_{p,t}$ which denotes the number of requests arriving at time t for page p. When we make a transmission of page p at time t using speed augmentation (this is one of the $\epsilon\delta\mathsf{opt}$ extra transmissions made at each grid time), we ensure that $B_p \mid n_{p,t}$ ($n_{p,t}$ is divisible by B_p) after the transmission. At each time, at most one transmission will be made using speed augmentation for each page. Thus we can assume that at each grid time we first make $\epsilon\mathsf{opt}$ 'regular' transmissions and then at most $\epsilon\delta\mathsf{opt}$ 'extra' transmissions. A regular transmission of a page is used to satisfy as many requests of the page as possible in FIFO fashion. In contrast, we use an extra transmission of page p only to have $B_p \mid n_{p,t}$.

We now describe how we find the extra pages we will transmit using speed augmentation. We will mark some pages at each grid time and will make an extra transmission of page p at time t if page p is marked at the time. To describe the marking procedure, we need to define W_p for each page p. We say that two requests ρ and ρ' for the same page p are adjacent if $r_\rho \ne r_{\rho'}$ and no requests arrive for page p between times r_ρ and $r_{\rho'}$. We can assume w.l.o.g. that any two adjacent requests ρ and ρ' for the same page arrive within opt time steps since otherwise, no transmission of page p can satisfy requests arriving no later than ρ and those arriving no earlier than ρ' simultaneously; here we assumed $r_\rho < r_{\rho'}$. For each page p, define $W_p := [\min\{r_\rho : p_\rho = p\}, \max\{r_\rho : p_\rho = p\} + \mathsf{opt}]$, which we call page p's window. Notice that we can assume w.l.o.g. that the optimal schedule transmits page p only during W_p. We are now ready to describe the marking procedure.

Marking procedure: Consider times in increasing order. There is a queue $\mathcal{Q}_t$ containing all pages p such that $t \in W_p$. Each page $p \in \mathcal{Q}_t$ is associated with its most recent marking time u_p. At each grid time t, $u_p \leftarrow t$ if the page has not been requested yet. At each grid time t, we choose up to $\delta\epsilon\mathsf{opt}$ pages with the smallest u_p and let $u_p \leftarrow t$ for such pages p.

Lemma 1. *Let $\tau_{p,t}$ denote the latest (grid) time no later than t when p is marked. For any time t and any page p such that $t \in W_p$, we have $t - 3\mathsf{opt}/\delta < \tau_{p,t} \le t$.*

Proof. For the sake of contradiction, let t^* be the first time t such that for some page p, $t \in W_p$ and page p is not marked at any time during $I = (t-3\mathsf{opt}/\delta, t]$. Note that we make $3\mathsf{opt}$ markings during I. A crucial observation is that no page is marked more than once during I since when a page q is marked, its latest marking time u_q will become greater than u_p and page p is not marked during $(t-3\mathsf{opt}/\delta, t]$. Let Q denote the set of pages marked during I. Note that $|Q| \ge 3\mathsf{opt}$. Let $t_1 := t - 3\mathsf{opt}/\delta$ be the start time of I. Note that for any $q \in Q$, $t_1 \in W_q$. This is because otherwise either q has not been requested till time t_1, thus u_q is greater than u_p, or all requests of page q must have been completed before time t_1. In either case, $q \notin Q$. Hence any page q in Q is requested during $[t_1 - \mathsf{opt}, t_1]$, meaning that all pages in Q must be transmitted during $[t_1 - \mathsf{opt}, t_1 + \mathsf{opt}]$ at least once. Since at most $(1+\delta)(2+\epsilon)\mathsf{opt}$ transmissions can be made during the interval and $|Q| \ge 3\mathsf{opt}$, we have a contradiction. □

The following lemma implies that there exists a $(1+\delta)$-speed schedule with a good structural property that is as good as the optimal schedule. The proof immediately follows by applying Lemma 1 to the optimal schedule.

Lemma 2. *There exists a $(1+\delta)$-speed schedule with the maximum flow time at most opt where for any p and t, if $n_{p,t} > 0$, then there is a time $t - 3\mathsf{opt}/\delta < t' \le t$ such that $B_p \mid n_{p,t'}$. Further, at each grid time t, the schedule makes an extra transmission of page p if and only if p is marked at the time t.*

As before, we count $|\mathcal{A}_t|$, the number of possibilities of A_t at each time t, which we can reduce using Lemma 2.

Lemma 3. *For any t, $|\mathcal{A}_t| \le (\frac{144}{\epsilon^2\delta^2}2^{7/\delta})^{\mathsf{opt}}$.*

Proof. Fix a time t and $p \in P(A_t)$. By Lemma 1, $\tau_{p,t}$ can have at most $\frac{3}{\epsilon\delta}$ different values. Let $\tau'_{p,t}$ be the latest time t' no later than t such that $B_p \mid n_{p,t'}$. Note that $\tau_{p,t} \le \tau'_{p,t}$. By definition of $\tau'_{p,t}$, we know that at least one request is alive for page p at each time during $(\tau'_{p,t}, t]$. This implies that every transmission of page p made during $(\tau'_{p,t}, t]$ satisfies exactly B_p requests, meaning that $n_{p,\tau'_{p,t}}$ and z_p, the number of transmissions of page p made during $(\tau'_{p,t}, t]$ determines $n_{p,t}$.

Let $N(\cdot)$ denote the number of possibilities of what is inside the parentheses. We first upper bound $N(P(A_t))$. Note that $P(A_t) \subseteq P(R_{t-\mathsf{opt}<r\le t})$, and $|P(R_{t-\mathsf{opt}<r\le t})| \le 2(1+\delta)\mathsf{opt} \le 4\mathsf{opt}$. Hence we have $N(P(A_t)) \le 2^{4\mathsf{opt}}$. Now we bound $N(\{\tau'_{p,t}\}_{p\in P(A_t)})$ for each fixed $P(A_t)$. As mentioned before, each $\tau'_{p,t}$ can have at most $\frac{3}{\epsilon\delta}$ different values. Hence, $N(\{\tau'_{p,t}\}_{p\in P(A_t)}) \le (\frac{3}{\epsilon\delta})^{2\mathsf{opt}}$ for each fixed $P(A_t)$; note that $|P(A_t)| \le (1+\delta)\mathsf{opt} \le 2\mathsf{opt}$ since all pages alive at time t must be transmitted at least once by time $t+\mathsf{opt}$. Since $\tau'_{p,t} \ge t - 3\mathsf{opt}/\delta$, we have that $\sum_{p\in P(A_t)} z_p \le 3(1+\delta)\mathsf{opt}/\delta \le 6\mathsf{opt}/\delta$. For each fixed $P(A_t)$, by Proposition 1, we have that $N(\{z_p\}_{p\in P(A_t)}) \le 2^{6\mathsf{opt}/\delta+2\mathsf{opt}} \le 2^{7\mathsf{opt}/\delta}$; recall that $\delta \le 1/10$. Hence $|\mathcal{A}_t| \le 2^{4\mathsf{opt}} \cdot (\frac{9}{\epsilon^2\delta^2})^{\mathsf{opt}} \cdot 2^{7\mathsf{opt}/\delta} = (\frac{144}{\epsilon^2\delta^2}2^{7/\delta})^{\mathsf{opt}}$. □

The DP used here is slightly different from the ones used in the previous sections since we have to decide the $\epsilon(1+\delta)$opt transmissions we make at each grid time. Let P_t denote the multi-set of pages we transmit by regular transmissions at time t. Let $t' = t - \epsilon$opt. Each entry DP(t, A_t) is set to true if and only if there are $A_{t'}$ and P_t such that A_t is achieved from $A_{t'} \cup R_{r=t}$ by transmitting pages in the multi-set P_t and rounding down $n_{p,t}$ to the nearest integer multiple of $B_{p,t}$ for each page p that is marked at time t. From the proof of Lemma 3, it is easy to see that the number of possibilities of P_t is upper bounded by $2^{4\text{opt}} \cdot 2^{7\text{opt}/\delta} \le 2^{8\text{opt}/\delta}$; here we did not optimize this loose bound. Therefore, we conclude that the running time of the DP is at most $(\frac{144}{\epsilon^2\delta^2} 2^{15/\delta})^{\text{opt}}$, in which small $poly(m)$ factors are omitted due to the asymptotic bound of $O(\frac{1}{\epsilon^3}\log T)$ on opt. An elementary calculation gives the running time claimed in Theorem 1 for the speed augmentation model.

3. Case opt $\geq (1/\epsilon^3)\log T$

We consider the following integer programming problem that determines if there is a feasible schedule with the maximum flow time of at most opt. Let $m_{p,I} := \sum_{t\in I} m_{p,t}$. See Section 2.4 for the definition of $m_{p,t}$ and W_p.

$$\sum_{t_1\le t\le t_2+\text{opt}} x_{p,t} \ge \lceil (m_{p,[t_1,t_2]}/B_p)\rceil \quad \forall p, t_1 \le t_2 \in [T] \tag{1}$$

$$\sum_{p} x_{p,t} \le 1 \qquad \forall t \in [T] \tag{2}$$

$$x_{p,t} = 0 \quad \forall p, t \notin W_p; \qquad x_{p,t} \in \{0,1\} \quad \forall p, t \in [T].$$

Lemma 4. *The above IP exactly captures the maximum flow time problem in batch scheduling. In other words, there exists a one-to-one mapping between a feasible solution to the IP and a feasible schedule with the maximum flow time at most* opt.

Proof. We will first discuss what each constraint captures. The first constraint ensures that all requests must be completed within opt time steps. The second constraint states that at most one page can be transmitted at each time. The third implies that no transmission is made if it cannot be used to satisfy a request within opt time steps.

The claimed one-to-one mapping is naturally defined from what $x_{p,t}$ is meant to encode: $x_{p,t} = 1$ if and only if page p is transmitted at time t, and requests for the same page are satisfied in FIFO order. To establish the one-to-one mapping, first consider an arbitrary schedule σ with the maximum flow time at most opt. Then, we know that to satisfy requests arriving during $[t_1, t_2]$ for page p, we need to transmit page p at least $\lceil m_{p,[t_1,t_2]}/B_p\rceil$ times during $[t_1, t_2+\text{opt}]$, which is exactly what the first constraint says. It is obvious that $\{x_{p,t}\}$ derived from σ satisfies other constraints.

We now show the opposite direction. Let $\{x_{p,t}\}$ be an arbitrary feasible solution to the IP and σ the schedule corresponding to the IP solution. We show that every request has flow time at most opt in σ. Let C_ρ denote the time when ρ is completed/satisfied. For the sake of contradiction, let ρ^* be a request with flow time greater than opt, that has the earliest completion time. Let $p = p_{\rho^*}$. Let t' be the latest time t before C_{ρ^*} such that a transmission of page p at time t satisfied less than B_p requests; if the time does not exist, let t' be the first time when a request arrives for page p, minus 1. Clearly, any request arriving by time t' is satisfied by time t' and has flow time at most opt by the definition of ρ^*. We use the first constraint with $t_1 = t' + 1$ and $t_2 = r_{\rho^*}$. We know that $m_{p,[t_1,t_2]}$ requests arrive during $[t_1, t_2]$ for page p, and we transmit page p during $[t_1, t_2+\text{opt}]$, $\lceil \frac{m_{p,[t_1,t_2]}}{B_p}\rceil$ times. Note that each transmission satisfies exactly B_p requests by the definition of t' and t_1. This means that ρ^* must be satisfied by time $t_2 + \text{opt}$, which is a contradiction to the assumption that ρ^* has flow time greater than opt. □

We relax the IP by replacing the last constraints with $x_{p,t} \ge 0$. Let us call the resulting LP as $\text{LP}_{\text{MaxFlow}}$. We solve $\text{LP}_{\text{MaxFlow}}$ and let $x^*_{p,t}$ denote the optimal fractional solution. As mentioned, we use the rounding scheme developed in [7]. For completeness, we summarize the rounding scheme and explain how it works for batch scheduling.

Pages are partitioned into groups $\mathcal{G}$ such that all pages in the same group $g \in \mathcal{G}$ have disjoint windows. It was shown in Lemma 2.2 in [7] that 2opt groups are enough for this partition using the structural property that any two 'adjacent' requests of a page p must be made within opt − 1 time steps. Here, we get a slightly looser bound, $|\mathcal{G}| \le 2\text{opt} + 1$ because in this paper we allow a request to be eligible to be satisfied upon its arrival, hence we can only have that any two 'adjacent' requests of a page p must be made within opt time steps; see the definition of W_p in Section 2.4. However, the remaining analysis in [7] remains exactly the same since constants were not optimized there.

For each group $g \in \mathcal{G}$, define a cumulative amount of transmission made by the fractional solution. Formally, define $y^*_{g,t} := \sum_{p\in g}\sum_{t'\le t} x^*_{p,t'}$. Now for each group g, pick α_g from $[0, 1]$ uniformly at random. We will obtain a tentative schedule σ_{temp} and transform it to σ_{final}. In σ_{temp}, all requests are satisfied within opt time steps, and more than one transmission can be made at a time. In σ_{temp}, for each group g, we transmit a page $p \in g$ at the earliest time t when $y^*_{g,t} - \alpha_g \ge k$ for each non-negative integer k. The grouping is done so that for any time t, there is at most one page $p \in g$ with $x^*_{p,t} > 0$. Then we transform σ_{temp} into the final feasible schedule σ_{final} by rescheduling transmissions in FIFO fashion. More precisely, we keep the order of transmissions, breaking ties arbitrarily, and ensure that every transmission should be made no earlier in σ_{final} than σ_{temp}. Also it is ensured that σ_{final} make at most one transmission at each time. We will show that the integral solution corresponding to σ_{final} is feasible to the IP with opt replaced with $(1+6\epsilon)$opt w.h.p., meaning that it represents a feasible schedule with the maximum flow time at most $(1+6\epsilon)$opt.

Proposition 2 immediately follows by observing that the LP solution x^* transmits page p by at least $\lceil \frac{m_{p,[t_1,t_2]}}{B_p}\rceil$ units during $[t_1, t_2 + \text{opt}]$, and the rounding scheme guarantees to make one transmission of page p before the fractional solution x^* accumulates another unit of transmission of the page p.

Proposition 2. *The IP solution corresponding to σ_{temp} satisfies the constraint (1).*

We complete the analysis by showing that w.h.p. no request's flow time increases too much in the transformation from σ_{temp} into σ_{final}. It was shown in [7] that the increase in the objective is upper-bounded by the global overflow, $\max_I OF(I)$ where $Q_{temp}(I)$ is the number of transmissions made in σ_{temp} during time interval I and $OF(I) := \max\{Q_{temp}(I) - |I|, 0\}$. In words, $OF(I)$ measures how many more transmissions are attempted to make during I than are allowed.

We now upper-bound the overflow during each fixed interval I as follows. The proof uses the facts that there are $O(\text{opt})$ random variables that contribute to the overflow, and each random variable corresponding to each group can contribute to the overflow $OF(I)$ by at most one. More precisely, for each group g, depending on the value of α_g, we make transmission of pages from g either $\lceil\sum_{t\in I}\sum_{p\in g} x^*_{p,t}\rceil$ or $\lfloor\sum_{t\in I}\sum_{p\in g} x^*_{p,t}\rfloor$ times during I—in the former case, g contributes to $OF(I)$ by $\lceil\sum_{t\in I}\sum_{p\in g} x^*_{p,t}\rceil - \sum_{t\in I}\sum_{p\in g} x^*_{p,t}$. Since we use the same rounding method, we get the same upper bound on the overflow which was shown in [7].

Lemma 5 ([7]). *Suppose that* $\text{opt} \ge \frac{1}{\epsilon^3}\log T$. *Then for any interval* I, $\Pr[OF(I) \ge 6\epsilon\text{opt}] \le 1/T^3$.

Finally, by applying a union bound over all possible intervals (at most T^2), we have that $\max_I OF(I)$ is at most $6\epsilon\mathsf{opt}$ w.h.p. As mentioned before, this implies that each transmission moves to the right by at most $6\epsilon\mathsf{opt}$ time steps from σ_{temp} to σ_{final}. This also implies that all the IP constraints are satisfied when opt is replaced with $(1+6\epsilon)\mathsf{opt}$. Thus, we have a randomized algorithm that yields a feasible schedule with the maximum flow time at most $(1+6\epsilon)\mathsf{opt}$ with probability least $1-1/T$. The algorithm can be easily derandomized using the method of pessimistic estimators [9,7]. By scaling ϵ, we have Theorem 1 when $\mathsf{opt} = \Omega(\frac{1}{\epsilon^3}\log T)$, with no resource augmentation. The running time is clearly polynomial in m, and is dominated by that of the DPs in Section 2.

Acknowledgments

This work was supported in part by NSF grants CCF-1218620, CCF-1409130, CCF-1433220, CCF-1541602, and CCF-1617653. The authors greatly thank the anonymous reviewer for the numerous comments which helped significantly to improve the presentation of this paper.

References

[1] N. Bansal, M. Charikar, R. Krishnaswamy, S. Li, Better algorithms and hardness for broadcast scheduling via a discrepancy approach, in: ACM-SIAM Symposium on Discrete Algorithms, 2014, pp. 55–71.
[2] A. Bar-Noy, S. Guha, Y. Katz, J.S. Naor, B. Schieber, H. Shachnai, Throughput maximization of real-time scheduling with batching, ACM Trans. Algorithms 5 (2) (2009) 18:1–18:17.
[3] J. Chang, T. Erlebach, R. Gailis, S. Khuller, Broadcast scheduling: Algorithms and complexity, ACM Trans. Algorithms 7 (4) (2011) 47.
[4] C. Chekuri, A. Gal, S. Im, S. Khuller, J. Li, R.M. McCutchen, B. Moseley, L. Raschid, New models and algorithms for throughput maximization in broadcast scheduling - (extended abstract), in: Workshop on Approximation and Online Algorithms, 2010, pp. 71–82.
[5] C. Chekuri, S. Im, B. Moseley, Online scheduling to minimize maximum response time and maximum delay factor, Theory Comput. 8 (1) (2012) 165–195.
[6] S. Im, B. Moseley, Brief announcement: online batch scheduling for flow objectives, in: ACM Symposium on Parallelism in Algorithms and Architectures, 2013, pp. 102–104.
[7] S. Im, M. Sviridenko, New approximations for broadcast scheduling via variants of α-point rounding, in: ACM-SIAM Symposium on Discrete Algorithms, 2015, pp. 1050–1069.
[8] B. Kalyanasundaram, K. Pruhs, Speed is as powerful as clairvoyance, J. ACM 47 (4) (2000) 617–643.
[9] P. Raghavan, Probabilistic construction of deterministic algorithms: Approximating packing integer programs, J. Comput. System Sci. 37 (2) (1988) 130–143.