ARTICLE IN PRESS IPL:3970

Information Processing Letters ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A note on on-line broadcast scheduling with deadlines

Xin Han [a,*], He Guo [b], Dawei Yin [a], Yong Zhang [a]

[a] Department of Computer Science, The University of Hong Kong, Hong Kong
[b] School of Software, Dalian University of Technology, China

ARTICLE INFO

Article history:
Received 19 March 2008
Received in revised form 4 July 2008

Communicated by C. Scheideler

Keywords:
Online algorithms
Broadcast scheduling
Competitive ratio

ABSTRACT

In this paper, we study an on-line broadcast scheduling problem with deadlines, in which the requests asking for the same page can be satisfied simultaneously by broadcasting this page, and every request is associated with a release time, deadline and a required page with a unit size. The objective is to maximize the number of requests satisfied by the schedule. In this paper, we focus on an important special case where all the requests have their spans (the difference between release time and deadline) less than 2. We give an optimal online algorithm, i.e., its competitive ratio matches the lower bound of the problem.

1. Introduction

Broadcasting technologies receive a lot attention on networks that employs broadcasting to disseminate data or information. In contract to the traditional point-to-point mode of communication, broadcasting technologies have an advantage that one broadcast by the server can simultaneously satisfy requests required from multiple clients for an identical message. In this paper, we focus a *pull-based* model of broadcast scheduling problems, which is formalized as below.

Problem description. There is a collection of pages $S = \{1, \ldots, n\}$, in the server. The clients send requests to ask for these pages and each request has a release time, deadline and a distinct page to ask for. The server answers requests by broadcasting pages. Note that a broadcast of a page can satisfy all the requests asking for the same page simultaneously and there is at most one page to be broadcasted at any time During broadcasting, the preemption is allowed, but if the broadcast of a page is preempted, then in case the server choose this page to broadcast again, it must from the start point not the break point, we call this as *preemption with restart*. When the request for the page that currently broadcast arrives it must be kept in the queue of unsatisfied requests.

In this paper, we consider each page has an unit size, i.e., any page can be broadcasted during one time unit, and the requests arrive over time. The scheduling algorithm used by the server has no knowledge of requests in advance and makes decisions only with information of requests having already arrived. We call this on-line broadcast scheduling. There are two models, *discrete* and *continuous* models. In discrete model, the arrival times and deadlines for all the requests are integral. For the online version of discrete model. Kim and Chwa [10] gave a best possible online algorithm with competitive ratio 2. Since new requests may arrive and end any time, the continuous model is more general, and is well-studied during these years. The main objectives are minimizing the flow time (response time) and maximizing the total throughput, i.e., the number of satisfied requests.

Previous results. Most of the previous works on on-line broadcast scheduling focus on minimizing the flow time [1–4,6–8,11]. On maximizing the throughput of on-line broadcast scheduling, Kim and Chwa [10] first gave a 5.828-competitive algorithm, then Chan et al. [5] showed that the competitive ratio of algorithm in [10] is at most 5. Recently, Zheng et al. [13] obtained a new on-line algo-

* Corresponding author.
E-mail addresses: xhan@cs.hku.hk (X. Han), guohe@dlut.edu.cn (H. Guo), dwyin@cs.hku.hk (D. Yin), yzhang@cs.hku.hk (Y. Zhang).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2008.10.002

Please cite this article in press as: X. Han et al., A note on on-line broadcast scheduling with deadlines, Information Processing Letters (2008), doi:10.1016/j.ipl.2008.10.002

rithm by looking forward two steps and proved that the competitive ratio is at most 4.56. The lower bound of maximizing the throughput of on-line broadcast scheduling is 4 which is from a related on-line interval scheduling problem [12]. Fung et al. [9] first studied this on-line broadcast scheduling problem with laxity (the span of a request, i.e., the difference between deadline and release time) constraints, and got some results as below. If all the requests have their laxity at least 2 then a nice and simple online algorithm with competitive ratio 2.618 is given. If all the requests have their laxity at most $\alpha < 2$, then an $f(\alpha)$-competitive algorithm can be achieved, where $4 < f(\alpha) \leqslant 4.714$. For off-line broadcast scheduling problem, maximizing the throughput is NP-hard [4].

Our contributions. In this paper, we focus on maximizing the throughput and give a 4-competitive algorithm if all the requests have laxity less than 2, which is optimal since the lower bound of this problem is also 4.

2. Preliminaries

A request R_i is defined as a triple (p_i, r_i, d_i), where p_i is the requested page, r_i and d_i are its release time and deadline, respectively.

Definition 1 *(Laxity).* For a request $R = (p, r, d)$, its laxity is defined as $(d - r)$.

Definition 2 *(Alive and dead).* Given a request $R = (p, r, d)$, if $(d - t) \geqslant 1$ then we say the request is alive at time t, otherwise, dead at time t.

For a request $R = (p, r, d)$, at time t its weight $W(R, t)$ is defined as the following table, i.e., if it is alive at time t then its weight is 1 otherwise 0.

$d - t$	$(-\infty, 1)$	$[1, +\infty)$
Weight	0	1

For a page P, its weight $W(P, t)$ is defined as the number of all requests alive at time t in a pending list, i.e.,

$$W(P, t) = \sum_{p_i = P} W(R_i, t).$$

Definition 3 *(Competitive ratio).* To evaluate an online algorithm, we use the standard measure called *competitive ratio*. For any input sequence L, let $A(L)$ be the cost by an online algorithm A and $OPT(L)$ be the cost by an optimal off-line algorithm. The *competitive ratio* of algorithm A is then defined as $R_A = \sup_L \frac{OPT(L)}{A(L)}$.

3. A tight upper bound for laxity less than 2

We first give an on-line algorithm then show that its competitive ratio is 4 which matches the lower bound [12]. Our algorithm is quite similar with ones in [5,10]. The main ideas of our algorithm are: (i) whenever we decide to broadcast a page, the page with the maximal weight is selected to be served; (ii) when a new request for page p_a arrives if to start broadcasting page p_a can double the profit (i.e., throughput), then we abort broadcasting the current page, and put the new request into a pending list and select a page with the maximal weight and broadcast that page (this is the difference between our algorithm and the ones in [5,10]). Otherwise, continue to broadcast the current page and put the new request in the pending list.

Let P_a be the page of a new request which arrives at the current time t, let P_c denote the currently broadcast page if it exists. Our algorithm is described (see Algorithm 1).

Algorithm 1. Weighting Pages (*WP*)

Initialize the profit W, i.e., $W \leftarrow 0$.
while (request-arrival or broadcast-completion) do
{
 request-arrival: Put new requests into the pending list.
if $W(P_a, t) \geqslant 2 \cdot W$ **then**
  Aborted page P_c, go to *selection* step (including the case $P_a = P_c$).
end if
broadcast-completion: Remove the requests satisfied, go to *selection* step if the pending list is not empty,
selection: Select a page P such that $W(P, t)$ is maximized (break a tie arbitrarily), and broadcast the page P and $W \leftarrow W(P, t)$, where t is the current time.
}

We first define a concept called *basic chain* and observe an important property related to it. Then, we divide the broadcasts by our algorithm into a set of basic chains and combine the property to get an upper bound 4 for the competitive ratio.

Definition 4 *(Basic chain).* For $i \leqslant j$, a sequence of broadcasts $(P_i, P_{i+1}, \ldots, P_j)$ is called a *basic chain* if pages $P_i, \ldots, P_{j-1}$ are aborted broadcasts and page P_j is a completed broadcast, and the broadcast just before P_i is empty or a completed broadcast.

Theorem 1. *For any input list of requests with laxity less than* 2*, the competitive ratio of our algorithm is* 4*.*

Proof. Let the sequence of broadcasts $(P_1, P_2, \ldots, P_j)$ be the first basic chain generated by our on-line algorithm. Let $(P_1^*, P_2^*, \ldots, P_m^*)$ be the first pages broadcast by an optimal scheduling such that the starting point of broadcasting page P_m^* is sat in the time interval for broadcasting page P_j, shown as Fig. 1 (if P_m^* does not exist, then we set P_m^* as a dummy page). Let time t_i (t_i^*) denote the starting point of broadcasting page P_i (P_i^*) for $1 \leqslant i \leqslant \max\{j, m\}$. Without loss of generality, assume that $(t_{i+1}^* - t_i^*) \geqslant 1$ for $1 \leqslant i \leqslant m$ otherwise we can get another optimal schedule by broadcasting P_h^* at time t_h^* except for page P_i^* and doing nothing during $[t_i^*, t_{i+1}^*)$.

Now, for $1 \leqslant i \leqslant j$, we define a set of intervals, $I_i = [t_i, t_{i+1})$, where $t_{j+1} = t_j + 1$. For an input list L, let $A(L)$ and $OPT(L)$ be the number of requests satisfied by our algorithm and an optimal schedule respectively. For $1 \leqslant i \leqslant m$, let x_i^* be the number of requests which are satisfied by broadcasting page P_i^* in the optimal schedule.

Please cite this article in press as: X. Han et al., A note on on-line broadcast scheduling with deadlines, Information Processing Letters (2008), doi:10.1016/j.ipl.2008.10.002

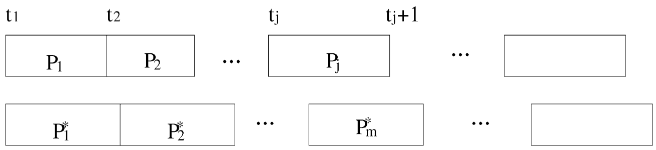

Fig. 1. Page P_m^* starts in time interval $[t_j, t_j+1)$.

Remember that the whole sequence of broadcasts $(P_1, P_2, \dots, P_j)$ by our algorithm is a basic chain. Then we have the following observation.

Claim 1. *For $1 \leqslant i \leqslant m$, $x_i^* \leqslant 2^{i+1-m}W$, where $W = W(P_j, t_j)$.*

Proof. Recall that t_i^* is the start time to broadcast page P_i^*. For $1 \leqslant i \leqslant m$, we have

$$\left(t_{i+1}^* - t_i^*\right) \geqslant 1. \tag{1}$$

On the other hand, the whole sequence of broadcasts $(P_1, P_2, \dots, P_j)$ by our algorithm is a basic chain, by the definition of a basic chain, for $1 \leqslant i \leqslant j$ we have

$$t_{i+1} - t_i \leqslant 1. \tag{2}$$

Let $I_f = [t_f, t_{f+1})$ be the interval such that $t_i^* \in I_f$. By (1) and (2), we have the number of broadcasts during $[t_f, t_j]$ by our algorithm is not less than the number by the optimal schedule during $[t_i^*, t_m^*]$, i.e.,

$$(j-f) \geqslant (m-i) \quad \Rightarrow \quad f \leqslant j-(m-i) = j+i-m. \tag{3}$$

According to algorithm *WP*, for any page P, we have

$$W(P, t_f) \leqslant 2^{f-j} W(P_j, t_j) = 2^{f-j} W. \tag{4}$$

Let t_i' be the time when the last request for page P_i^* released at or before time t_i^* in the input list. Let $t_i'' = \max\{t_f, t_i'\}$. According to algorithm *WP*, at time t_i'' there is a comparison between $W(P_i^*, t_i'')$ and $W(P_f, t_f)$. Since page P_f is broadcast at time t_i'' by our algorithm, we have

$$W\left(P_i^*, t_i''\right) < 2W(P_f, t_f). \tag{5}$$

By inequalities (3)–(5), we have

$$W\left(P_i^*, t_i''\right) \leqslant 2^{i+1-m} W. \tag{6}$$

Let x_i'' be the number of requests for page P_i^* which are alive at time t_i''. By the definition of x_i'' and $t_i'' \leqslant t_i^*$, we have

$$x_i^* \leqslant x_i''. \tag{7}$$

By (7) and (6)

$$x_i^* \leqslant W\left(P_i^*, t_i''\right) \leqslant 2^{i+1-m} W. \qquad \square \tag{8}$$

Next we are going to bound $OPT(L)$, i.e., the throughout by the optimal schedule. Without loss of generality, broadcasts generated by our algorithm and the optimal algorithm looks like Fig. 1.

We prove this theorem by mathematical induction over the number of basic chains produced by our algorithm. First, we prove that the theorem holds if the whole broadcast generated by algorithm *WP* is a basic chain. Then we assume that the theorem holds for the case in which there are h basic chains. Finally, we prove that the theorem still holds for $(h+1)$ basic chains.

Step 1. There is only one basic chain in the whole broadcast generated by our algorithm. In this case, we prove P_m^* is the last page in the optimal schedule. Otherwise there is at least one request alive at time t_m^*+1, where $t_m^* \geqslant t_j$. Since all the requests have laxity less than 2, the alive request must be released after t_j. So, the request would have been broadcasted by our algorithm after t_j+1. But, this contradicts with the fact that the whole sequence of broadcasts $(P_1, P_2, \dots, P_j)$ generated by our algorithm is one basic chain. So page P_m^* is the last page in the optimal schedule. By Claim 1, we have

$$OPT(L) = \sum_{i=1}^{m} x_i^* \leqslant \sum_{i=1}^{m} 2^{i+1-m} W$$
$$= \frac{4W}{2^m} \sum_{i=0}^{m-1} 2^i \leqslant 4W = 4A(L).$$

Step 2. Assume that this theorem holds when there are h basic chains in the whole broadcast generated by our algorithm, where $h \geqslant 1$. Next we consider the case in which there are $h+1$ basic chains in our broadcast. First, we define four sublists of requests. We define L_2 as the sublist of requests that will be considered by our algorithm after time t_j+1, i.e., the sublist of requests with release time at least t_j+1 or requests which are still alive at time t_j+1 (t_m^*+1) and not satisfied by our algorithm before t_j+1. In the same way, we define L_2^* as the sublist of requests that will be considered by the optimal schedule after t_m^*+1, i.e., L_2^* is the sublist of requests with release time at least t_m^*+1 or requests which are still alive at time t_m^*+1 and not satisfied by the optimal algorithm before time t_m^*+1. Let $L_1 = L - L_2$ and $L_1^* = L - L_2^*$. By definitions, L_1 is the sublist of requests with release time before t_j+1 and not alive at time t_j+1, i.e, requests with release time before t_j+1 and not satisfied, or requests satisfied before t_j+1. Observe that for any request $R \in L_1$, if request R is not satisfied by algorithm *WP*, then request R is not alive at time t_j+1, therefore R is not alive at time t_m^*+1 too, where $t_m^* \geqslant t_j$. Then $R \notin L_2^*$. If request R is satisfied by algorithm *WP* before t_j+1, then it is not alive at time t_m^*+1 since every request has laxity less than 2. So, in both cases, we have $R \notin L_2^*$, where $R \in L_1$. Then

$$L_2^* \subseteq L_2 = L - L_1. \tag{9}$$

To estimate the throughput by algorithm *WP* and the optimal algorithm, we need to modify L_2 and L_2^* slightly. For every request in L_2 $(\in L_2^*)$, if its release time is at least

Please cite this article in press as: X. Han et al., A note on on-line broadcast scheduling with deadlines, Information Processing Letters (2008), doi:10.1016/j.ipl.2008.10.002

$t_j + 1$ then just copies it into L_2^1 (L_2^{*1}) else modifies its release time to $t_j + 1$ then copies it into L_2^1 (L_2^{*1}). For every request in L_2^*, if its release time is at least $t_m^* + 1$ then just copies it into L_2^{*2} else modifies its release time to $t_m^* + 1$ then copies it into L_2^{*2}. By the above definitions and (9), we have

$$OPT(L_2^1) \geqslant OPT(L_2^{*1}) \geqslant OPT(L_2^{*2}), \tag{10}$$

$$A(L) = A(L_1) + A(L_2^1), \tag{11}$$

$$OPT(L) = OPT(L_1^*) + OPT(L_2^{*2}). \tag{12}$$

And by the assumption for h basic chains and Claim 1, we have

$$4A(L_2^1) \geqslant OPT(L_2^1) \quad \text{and} \quad 4A(L_1) \geqslant OPT(L_1^*). \tag{13}$$

So,

$$\begin{aligned} 4A(L) &= 4(A(L_1) + A(L_2^1)) \quad \text{by (11)} \\ &\geqslant OPT(L_1^*) + OPT(L_2^1) \quad \text{by (13)} \\ &\geqslant OPT(L_1^*) + OPT(L_2^{2*}) \quad \text{by (10)} \\ &= OPT(L) \quad \text{by (12)} \end{aligned}$$

Hence, this theorem holds. □

References

[1] N. Bansal, M. Charikar, S. Khanna, J. Naor, Approximating the average response time in broadcast scheduling, in: SODA, 2005, pp. 215–221.
[2] N. Bansal, D. Coppersmith, M. Sviridenko, Improved approximation algorithms for broadcast scheduling, in: SODA 2006, pp. 344–353.
[3] Y. Bartal, S. Muthukrishnan, Minimizing maximum response time in scheduling broadcasts, in: SODA 2000, pp. 558–559.
[4] J. Chang, T. Erlebach, R. Gailis, S. Khuller, Broadcast scheduling: algorithms and complexity, in: SODA 2008.
[5] W. Chan, T. Lam, H. Ting, P. Wong, New results on on-demand broadcasting with deadline via job scheduling with cancellation, in: COCOON 2004, pp. 210–218.
[6] J. Edmonds, Scheduling in the dark, Theoretical Computer Science 235 (1) (2000) 109–141.
[7] J. Edmonds, K. Pruhs, Multicast pull scheduling: when fairness is fine, in: SODA 2002, Algorithmica 36 (2003) 315–330.
[8] J. Edmonds, K. Pruhs, A maiden analysis of longest wait first, in: SODA 2004, ACM Transactions on Algorithms 1 (1) (2005) 14–32.
[9] S.P.Y. Fung, F.Y.L. Chin, C.K. Poon, Laxity helps in broadcast scheduling, in: Proceedings of 9th Italian Conference on Theoretical Computer Science, 2005, pp. 251–264.
[10] J. Kim, K. Chwa, Scheduling broadcasts with deadlines, Theoretical Computer Science 325 (2004) 479–488.
[11] J. Robert, N. Schabanel, Pull-Based data broadcast with dependencies: be fair to users, not to items, in: SODA 2007, pp. 238–247.
[12] G.J. Woeginger, On-line scheduling of jobs with fixed start and end times, Theoretical Computer Science 30 (1994) 5–16.
[13] S. Fung, F. Zheng, W. Chan, F. Chin, C. Poon, P. Wong, Improved online broadcast scheduling with deadlines, Journal of Scheduling 11 (2008) 299–308.

Please cite this article in press as: X. Han et al., A note on on-line broadcast scheduling with deadlines, Information Processing Letters (2008), doi:10.1016/j.ipl.2008.10.002