Online Broadcast Scheduling: Minimizing the Maximum Response Time

Jessica Chang*

Abstract

Broadcast scheduling is a widely used mechanism for disseminating vast amounts of information. In this model, information is discretized into "pages", which clients explicitly request. The primary advantage of broadcasting is that any number of outstanding requests may be satisfied via only one broadcast. There are several ways to measure the quality of a schedule. The majority of this paper is devoted to the study of the FIFO algorithm for online broadcasting, in which the server does not know anything about future requests at any given time, with the goal of minimizing the maximum response time. In addition, an integrality gap for throughput maximization and a few greedy approaches to the problem of broadcasting with fixed maximum response times are described.

1 Introduction

Scheduling scenarios typically involve finding a way to satisfy numerous independent requests for resources. In traditional scheduling, several jobs may arrive over a period of time, each job with an independent deadline. The scheduler's job is to schedule these jobs in such

*Research completed under the advisement of Dr. Samir Khuller and supported by a Senior Summer Scholar award from the University of Maryland and by NSF REU Supplement to Award CCR-0430650. Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : jschang@umd.edu.

a way to create a 'satisfactory' schedule. In unicast scheduling, one job may be processed at any given time. In multicast scheduling, several (but a limited number of) jobs may be scheduled simultaneously.

Broadcast scheduling is a mechanism in which information is disseminated by means of a server (e.g. radio or satellite) to a large number of users. In this model, clients request information rather than the resources to complete a specific job. The information is discretized into n discrete pieces, denoted as "pages" 1 to n and can be relayed to any number of users via one broadcast, as opposed to individual and redundant information transfers. Time is broken into discrete intervals, called "slots". Time slot t will refer to the unit of time between $t-1$ and t and will henceforth be denoted as time t where the context makes the meaning unambiguous. At each time t, the server is permitted to broadcast exactly one page.

Each user issues a request for a particular page and waits for that request to be satisfied. When we say that a user has requested a page at time t, we mean that the user requests the page at the precise moment before time slot t begins. For instance, if the page scheduled at time t is also the page requested at time t, then that request is considered to be satisfied at time t. Once a page is broadcast, all outstanding requests for that page are satisfied. This is the primary advantage over traditional scheduling, in which at most one request (or a fixed number of requests) can be satisfied at a time.

This model becomes increasingly relevant as the demand for information continues to grow exponentially with respect to time and service providers; information providers are seeking ways to meet this demand without overloading their servers. Already, broadcasting models have been implemented in commercial systems including the Hughes DirecPC System [5] and the Intel Intercast System [9].

There are several ways to measure the quality of a broadcast schedule. Intuitively, the "response time" for a request is the duration of time which that request has to wait for its page to be scheduled. One common measure of quality is the sum (or, equivalently, the average) of response times over all requests: the best schedule is that which minimizes this this sum. Another measure, the maximum response time, is the primary focus of this paper.

1.1 Related Work

The problem of minimizing the average response time has been proven by Erlebach and Hall to be NP-complete[1] [6]. Gandhi et al. found a 2-speed 2-approximation [7] for this objective function. A k-speed algorithm can broadcast up to k pages simultaneously and an α-approximation algorithm for a minimization problem is guaranteed to perform no worse than α times the optimal solution. Bartal et al. provided an algorithm that runs faster, but does not yield a better approximation bound [2].

Another objective function is to minimize the maximum response time. It is known that this problem is NP-complete [3], and a 2-approximation for the offline version (in which all requests are known before any decisions are made about which pages to broadcast) was provided in [2]. In this paper, we consider the online version, where at any time t, the server has no knowledge of future requests when it makes a decision about what to schedule at time t.

Another well-studied problem is throughput maximization, in which each request for a page comes with a pre-defined deadline. The goal is to maximize the number of requests satisfied before their given deadline. Bar-Noy et al. [1] gave a $\frac{1}{2}$-approximation for this problem, i.e. if the optimal solution satisfies M requests, their algorithm is guaranteed to satisfy at least $\frac{1}{2}M$ requests. This approximation was later improved to $\frac{3}{4}$ by Gandhi et al. [8] and more recently, was proven to be NP-complete [3].

1.2 Problem Definition

We formally define the broadcasting scenario as follows: there are n possible pages, $P = \{1, 2, \ldots, n\}$. We assume that time is discrete and at time t, any subset of pages can be requested. Let (p, t) represent a request for page p at time t. Let r_t^p denote number of requests (p, t). A time slot t is the window of time between time $t-1$ and time t. The

[1] A problem is NP-complete means that it cannot be solved in polynomial time unless all known NP-complete problems can also be solved in polynomial time. It is generally accepted that NP-complete problems cannot be solved quickly, i.e. in polynomial time, and much research has been devoted to approximating the solutions to these problems.

server can broadcast a page in each time slot. When a page is broadcast in time-slot t, we will simply say that it has been broadcast at time t. We say that a request (p, t) is satisfied at time S_t^p, if S_t^p is the first time instance *after* t when page p is broadcast. The response time of request (p, t) is $S_t^p - t$.

1.3 Summary of Results

The majority of this paper is devoted to the analysis of the online broadcasting model for minimizing the maximum response time. We show that the $FIFO$ algorithm yields a 2-approximation and that there exists no online deterministic $(2 - \epsilon)$-approximation for any $\epsilon > 0$. In addition, some greedy approaches are described as approximations for the problem of maximizing the number of satisfied requests given a fixed maximum response time, though none of these approaches is known to perform better than the current bound [8]. Finally, we show that for throughput maximization, the LP-rounding technique cannot approximate the optimal solution by more than $\frac{17}{18}$. This shows that the LP-rounding approach cannot lend approximations that are arbitrarily close to the optimal solution, despite the fact that the best known algorithm to date uses this technique [8].

2 Analysis of FIFO

The $FIFO$ algorithm (First-In-First-Out) schedules pages in the order that they are requested. If requests for multiple distinct pages arrive at the same time, $FIFO$ schedules these pages in arbitrary order. We define the 'queue' of a schedule at time t to be the set of pages for which outstanding requests exist at time t.

For example, suppose that at time $t = 0$, pages A and B are requested, and at $t = 1$, B and C are requested. One schedule may broadcast pages in the following order: [B, A, B, C], with the maximum response time as 3 (for page C). Note that after page B has been first broadcast, the queue contains only page A. After B and C are requested again at $t = 1$, the queue contains pages A, B and C. A better schedule would broadcast page A first so that both requests for page B can be satisfied via one broadcast. The optimal schedule would be [A, B, C] with a maximum response time of 2. The maximum size of

the queue over the entire schedule is 2. Note that the length of the queue at any time is a lowerbound on the performance of the entire schedule.

We will prove that $FIFO$ is 2-competitive and that no deterministic online algorithm can perform better. These claims were made in [2], but their construction for the former claim was incorrect and no proof was available for the latter.

Theorem 1. *For any constant $\epsilon > 0$, there exists no $(2-\epsilon)$-approximation for the online model of minimizing the maximum response time.*

Proof. Consider the following example. Let the client request pages $1, 2, \ldots, n$ at time $t = 0$, and then request whatever the online algorithm broadcasts immediately after that page is broadcast. This is done for a total of $n-1$ steps. At the point of this last request, we see that OPT has nothing in its queue while the online algorithm has $n-1$ outstanding requests. Then, suppose that at time $t = n-1$, the client requests a set of n new pages, say pages $n+1, \ldots, 2n$. OPT can schedule these items in the next n timeslots, but the online algorithm now has n new pending requests in addition to the $n-1$ pages that were already in its queue. Even in the best case where the online algorithm continues to schedule what was originally in its queue, the requests for the new n items will not begin to be satisfied until $n-1$ time units after the requests were made. Therefore, the last page scheduled by the online algorithm would have waited a total of $2n-1$ time slots, while OPT would have scheduled pages $n+1, \ldots, 2n$ within a maximum response time of n. The competitive ratio, $\frac{2n-1}{n}$, is arbitrarily close to 2. □

Theorem 2. *FIFO is 2-competitive online algorithm.*

Proof. Let us denote $FIFO$'s and OPT's queues at time t as F_t and O_t respectively. Since the particular advantage of broadcast scheduling over unicast scheduling is the ability to satisfy more than one request at a time, we shall refer to such occasions as the "merging" of requests. In other words, the merging of two requests (p, i) and (p, j), for $i \neq j$, refers to the occasion where $S_i^p = S_j^p$.

Consider the difference between $FIFO$'s queue and OPT's queue, denoted as $F_t \backslash O_t$. If, at all times t, $|F_t \backslash O_t|$ is upper bounded by OPT, then the claim follows easily, since

$|F_t \cap O_t| \leq |O_t| \leq OPT$. Suppose that $|F_t \backslash O_t|$ first exceeds OPT at time t^*. We derive the following contradiction: if $|F_{t^*} \backslash O_{t^*}| > OPT$, then $|F_{t^*-\delta} \backslash O_{t^*-\delta}| > OPT$ for some earlier time $t^* - \delta$.

Let P be the set of pages in $F_{t^*} \backslash O_{t^*}$. In general, a page p_i may have several outstanding requests at a time t. Since their response times cannot exceed that of the earliest outstanding request for p_i, these subsequent requests can be ignored without changing the performance of the schedule. Thus, without loss of generality, we can assume that there is exactly one outstanding request at time t^* for each page in P. Let the pages of P be ordered by the time of their request, i.e. $P = \{p_1, \ldots, p_k\}$ such that the time of p_i's request t_i is before or equal to the time of p_j's request t_j whenever $i < j$.

Let $t^* - \delta$ be the time that the request for page p_1 arrives, and define interval W as the interval $[t^* - \delta,\ t^*]$. Note that $\delta \geq k > OPT$, since OPT satisfies all k requests for the distinct p_i's in interval W, but may schedule other pages during that time also. At the same time, $FIFO$ satisfies no requests of P during interval W since these pages are still in $FIFO$'s queue at time t^*. Let the set of pages scheduled by $FIFO$ during interval W be $Q = \{q_1, \ldots, q_\delta\}$, ordered by the time in which they were scheduled. All of these pages were in $FIFO$'s queue at time $t^* - \delta$ since we know that $FIFO$ never scheduled p_1 in interval W. Therefore, all pages in Q must be distinct.

We want to prove that at most $(\delta - k)$ of these pages are also in OPT's queue at time $t^* - \delta$, i.e., OPT can schedule at most $(\delta - k)$ of these pages in interval W. (Note that given the length of window W, OPT cannot satisfy requests for the pages of Q after time t^*, as this would increase the maximum response time beyond OPT.) If OPT does not merge requests from Q and P, then it follows that OPT can schedule at most $(\delta - k)$ pages of Q in interval W. Then, OPT must schedule the rest before time $t^* - \delta$, and $|F_{t^*-\delta} \backslash O_{t^*-\delta}| \geq k > OPT$. □

Lemma 1. *OPT does not merge outstanding requests from sets P and Q.*

Proof. Suppose OPT merges outstanding requests from set Q and set P in the interval W. Consider the last merge, between the request(s) satisfied by $FIFO$ in broadcasting page q_i, and the request not satisfied by $FIFO$ in interval W for page p_ℓ (scheduled at some time

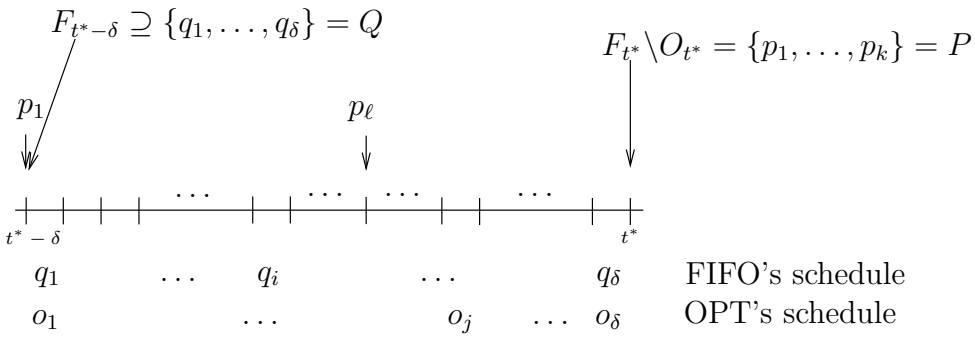

Figure 1: OPT's and FIFO's schedules for interval W

$t^* - \delta + j$, $j > i$). Let Q' be the set of pages $\{q_{i+1}, \ldots, q_\delta\}$. By assumption, no pages of Q' can be merged with pages of P by OPT. Therefore, OPT can schedule at most $(\delta - k)$ pages of Q' in interval W since OPT is busy scheduling the pages of P during the other k time slots. Then, at least $(\delta - i) - (\delta - k) = (k - i)$ pages of Q' were scheduled by OPT before interval W. Because of the nature of $FIFO$, everything in Q' was scheduled after page q_i was requested, so the wait in OPT's schedule for q_i is at least $k - i + j > k > OPT$. This is a contradiction. □

3 Scheduling with Fixed Maximum Response Times

Consider the following variation of the broadcast scheduling problem: given that a request may only be satisfied within T of its arrival, the goal is to create a schedule that maximizes the number of requests satisfied. This is a restricted form of the throughput maximization problem, since the problem can be re-defined as all requests (p_i, t_i) having deadlines of $t_i + T$. Thus throughput maximization $\frac{3}{4}$ approximation applies [8]. However, this problem is more specific than throughput maximization and the added structure may allow us to improve the current bound. Some of the greedy approaches explored are described in this section.

In an initial attempt to attain a approximation bound, a simple greedy algorithm was explored, but did not yield a ratio comparable to existing approximations. The general Maximum Coverage problem has a greedy $(1 - \frac{1}{e})$-approximation algorithm and can be applied in this broadcasting context.

Chekuri et al. provide a 2-approximation for Maximum Coverage with Groups(MCG) [4]. Given a set of elements S, and a set of subsets of S where each subset belongs to exactly one group, the goal is to pick at most one subset from each group, such that the number of elements covered is maximized. In other words, maximize the size of the set $M = \{e \in S | \ e \in (S_{g_1} \cup S_{g_2} \cup \ldots \cup S_{g_k})\}$, where S_{g_i} is the subset picked from group g_i. MCG is actually a relaxation of this broadcast scheduling problem: we can look at each time slot t as a group g_t. At each time slot, we have a number of outstanding requests that are still able to be satisfied; these requests correspond to the subsets. The distinct pages of information are the elements of the set, and may be part of more than one subset, since there may be multiple outstanding requests for a page at any given time. The algorithm proposed in [4] yields a $\frac{1}{2}$-approximation and we were unable to improve this bound for the broadcast scheduling problem.

4 Integrality Gap for Throughput Maximization

The best known approximation for the throughput maximization problem uses an LP-rounding technique. First, the problem is redefined as a linear program, for which optimal solutions may be fractional and can be found in polynomial time. However, in the original problem we do not allow the server to schedule a fraction of a page during one time slot. (At any given time t, a page is either broadcast or not broadcast.) So the approximation rounds the fractional solution into a valid integral one. We show by the following example that no LP-rounding approach can yield an approximation ratio better than $\frac{17}{18}$.

Three pages A, B and C are requested repeatedly in the following way: at each time t, page A is requested with deadline $t+2$, page B is requested with deadline $t+3$, and page C is requested with deadline $t+6$. It is not difficult to see that the optimal fractional solution satisfies all requests before their given deadlines: in each time slot, schedule $\frac{1}{2}$ of page A, $\frac{1}{3}$ of page B and $\frac{1}{6}$ of page C.

The optimal integral solution must lose at least 1 request for every 18 requests by the following argument: if a schedule broadcasts α pages of C, at most 6α requests for C can be satisfied. At the same time, for every page C scheduled, at least one request for page A

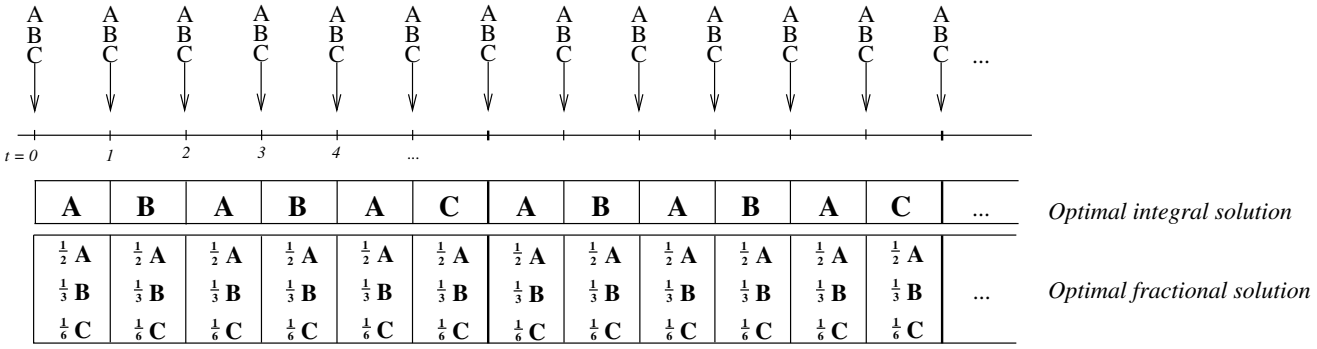

Figure 2: Optimum fractional and integral solutions for throughput maximization instance

or B is not satisfied. This is true, even for pages C scheduled close to each other.

Consider a time interval of length $6T$. Any schedule that broadcasts fewer than T pages for C, say $T - \beta$ pages, will lose at least $T + 5\beta > T$ requests. (The schedule can satisfy at most $6(T - \beta)$ requests for C, and so will lose at least 6β requests for C, in addition to the $T - \beta$ requests lost for either A or B.) However, any schedule that broadcasts more than T pages of C will lose more than T requests for A or B. All schedules that broadcast exactly T pages of C lose at least T of the $18T$ requests. Thus, all schedules must lose at least T of the $18T$ requests, regardless of how many times page C is scheduled. The schedule that repeatedly broadcasts [A, B, A, B, A, C] loses exactly T of $18T$ requests and is optimal.

Many thanks to Dr. Samir Khuller for his invaluable guidance and insight. This work was completed at the University of Maryland, College Park as part of the Department of Computer Science honors thesis requirement. It has been submitted for academic credit (course number: CMSC390).

References

[1] A. Bar-Noy, S. Guha, Y. Katz, J. Naor, B. Schieber and H. Schachnai. Throughput Maximization of Real-Time Scheduling with Batching. In *Proc. of 13th ACM-SIAM Symposium on Discrete Algorithms* (2002), pp. 742-751.

[2] Y. Bartal and S. Muthukrishnan. Minimizing maximum response time in scheduling broadcasts. In *Proc. of 11th ACM-SIAM Symposium on Discrete Algorithms* (2000), pp. 558-559.

[3] J. Chang, T. Erlebach, R. Gailis and S. Khuller. Broadcast scheduling: algorithms and complexity. *Manuscript* (2007).

[4] C. Chekuri and A. Kumar. Maximum Coverage Problem with Group Budget Constraints and Applications. In *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems* , vol. 3211 of *Lecture Notes in Computer Science*, Springer, pp. 72-83, 2004.

[5] DirecPC website, http://www.direcpc.com

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

[7] R. Gandhi, S. Khuller, Y. Kim and Y.C. Wan. Algorithms for minimizing response time in broadcast scheduling. In *Proc. of 9th Conference on Integer Programming and Combinatorial Optimization* (2002) and *Algorithmica* Vol. 38: 597-608 (2004).

[8] R. Gandhi, S. Khuller, S. Parthasarthy and A. Srinivasan. Dependent rounding in bipartite graphs. In *Proc. of 43rd Foundations of Computer Science Conference* (2002), pp. 323-332.

[9] Intel intercast website, http://www.intercast.com