Algorithms for Minimizing Response Time in Broadcast Scheduling*

Rajiv Gandhi[1], Samir Khuller[2], Yoo-Ah Kim[1], and Yung-Chun (Justin) Wan[1]

[1] Department of Computer Science, University of Maryland, College Park, MD 20742. {gandhi,ykim,ycwan}@cs.umd.edu

[2] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. samir@cs.umd.edu

Abstract. In this paper we study the following problem. There are n pages which clients can request at any time. The arrival times of requests for pages are known. Several requests for the same page may arrive at different times. There is a server that needs to compute a good broadcast schedule. Outputting a page satisfies all outstanding requests for the page. The goal is to minimize the average waiting time of a client. This problem has recently been shown to be NP-hard. For any fixed α, $0 < \alpha \leq \frac{1}{2}$, we give a $\frac{1}{\alpha}$- speed, polynomial time algorithm with an approximation ratio of $\frac{1}{1-\alpha}$. For example, setting $\alpha = \frac{1}{2}$ gives a 2-speed, 2-approximation algorithm. In addition, we give a 4-speed, 1-approximation algorithm improving the previous bound of 6-speed, 1-approximation algorithm.

1 Introduction

There has been a lot of interest lately in data dissemination services, where clients request information from a source. Advances in networking and the need to provide data to mobile and wired devices have led to the development of large-scale data dissemination applications (election results, stock market information etc). While the WWW provides a platform for developing these applications, it is hard to provide a completely scalable solution. Hence researchers have been focusing their attention on Data Broadcasting methods.

Broadcasting is an appropriate mechanism to disseminate data since multiple clients can have their requests satisfied simultaneously. A large amount of work in the database and algorithms literature has focused on scheduling problems based on a broadcasting model (including several PhD theses from Maryland and Brown) [7, 9, 4, 1, 5, 2, 8, 23, 6]. Broadcasting is used in commercial systems, including the Intel Intercast System [15], and the Hughes DirecPC [13]. There are two primary kinds of models that have been studied – the first kind is a *push*-based scheme, where some assumptions are made on the access probability for a certain data item and a broadcast schedule is generated [3, 9, 7, 17, 6]. We focus our attention on the second kind, namely *pull-based* schemes, where clients

* Research supported by NSF Awards CCR-9820965 and NSF CCR-0113192.

request the data that they need (for example via phone lines) and the data is delivered on a fast broadcast medium (often using satellites) [5]. This model is motivated by wireless web applications. This work deals entirely with the *pull-based* model, where requests for data arrive over time and a good broadcast schedule needs to be created.

A key consideration is the design of a good broadcast schedule. The challenge is in designing an algorithm that generates a schedule that provides good average response time. Several different scheduling policies have been proposed for on-line data broadcast. Aksoy and Franklin [5] proposed one such online algorithm. Their algorithm, called RxW, takes the product of the number of outstanding requests for page with the longest waiting time to compute the "demand" for a page. The page with the maximum demand is then broadcast. Another reasonable heuristic is to take the sum of waiting times of all outstanding requests for a page to compute its demand. Tarjan et al [22] have recently shown how to implement this algorithm efficiently. Not surprisingly, the worst case competitive ratio of these algorithms is unbounded, as shown in [18].

While the practical problem is clearly online, it is interesting to study the complexity of the offline problem as well. In trying to evaluate the performance of online algorithms, it is useful to compare them to an optimal offline solution. In addition, when the demands are known for a small window of time into the future (also called the look-ahead model in online algorithms) being able to quickly compute an optimal offline solution can be extremely useful. Many kinds of demands for data (e.g., web traffic) exhibit good predictability over the short term, and thus knowledge of requests in the immediate future leads to a situation where one is trying to compute a good offline solution.

One could also view the requests in the offline problem as release times of jobs, and one is interested in minimizing average (weighted) flow time. While this problem has been very well studied (see [11] and references therein), the crucial difference between our problem and the problem that has been studied is the fact that scheduling a job satisfies *many* requests simultaneously. (The term "overlapping jobs" has also been used to describe such scheduling problems in the past.)

The informal description of the problem is as follows. There are n data items, $1, \ldots, n$, called pages. Time is broken into "slots". A time slot is defined as the unit of time to transmit one page on the wireless channel. A request for a page j arrives at time t and then waits. When page j has been transmitted, this request has been satisfied. Arrival times of requests for pages is known, and we wish to find a broadcast schedule that *minimizes the average waiting time.* There has been a lot of work on this problem when we assume some knowledge of a probability distribution for the demand of each page [6, 7, 17].

In the same model, the paper by Bartal and Muthukrishnan [8] studies the problem of minimizing the maximum response time. The offline version has a PTAS, and there is a 2-competitive algorithm for the online version [8]. (They also credit Charikar, Khanna, Motwani and Naor for some of these results that were obtained independently.)

The recent paper by Kalyanasundaram et al. [18] studies this problem as well. They showed that for any fixed $\epsilon, 0 < \epsilon \leq \frac{1}{3}$, it is possible to obtain a $\frac{1}{\epsilon}$-speed $\frac{1}{1-2\epsilon}$-approximation algorithm for minimizing the average response time, where a k-speed algorithm is one where the server is allowed to broadcast k pages in each time slot. For example by putting $\epsilon = \frac{1}{3}$ they obtain a 3-speed, 3-approximation. The approximation factor bounds the cost of the k-speed solution compared to the cost of an optimal 1-speed solution. (This kind of approximation guarantee is also referred to as a "bicriteria" bound in many papers.) Note that we cannot set $\epsilon = \frac{1}{2}$ to get a 2-speed, constant approximation. Their algorithm is based on rounding a fractional solution for a "network-flow" like problem that is obtained from an integer programming formulation. This problem has recently shown to be NP-hard by Erlebach and Hall [14].

Our Results: We consider a different Integer Program(IP) for this problem, and show that by relaxing this IP, for any fixed $\alpha \in (0, \frac{1}{2}]$, we can obtain a $\frac{1}{\alpha}$-speed solution that is a $\frac{1}{1-\alpha}$-approximation. For example, by setting $\alpha = \frac{1}{2}$ we obtain a 2-speed, 2-approximation. By setting $\alpha = \frac{1}{3}$ we obtain a 3-speed, 1.5-approximation. Note that our algorithm improves both the speed and approximation guarantee of their algorithm [18]. This can be viewed as a step towards ultimately obtaining a 1-speed algorithm. The rounding method is of independent interest and quite different from the rounding method proposed by Kalyanasundaram et al. [18]. Moreover, our formulation draws on rounding methods developed for the k-medians problem [10]. This connection with scheduling is perhaps a surprising aspect of our work. Erlebach and Hall [14] showed that one can get a 1-approximation via a 6-speed algorithm. Here we show how to use the method in this paper to get a 1-approximation via a 4-speed algorithm.

2 Problem

The problem is formally stated in [18], but for the sake of completeness we will describe it here. 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). Let T be the time of last request for a page. Without loss of generality, we can assume that T is polynomially bounded. A time slot t is the window of time between time $t-1$ and time t. We also have a k-speed server that can broadcast up to k pages at any 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. In this paper, we work in the offline setting in which the server is aware of all the future requests. Our goal is to schedule the broadcast of pages in a way so as to minimize the total response time of all requests. The total response time is given by

$$\sum_p \sum_t r_t^p (S_t^p - t) \ .$$

Consider the example shown in Fig. 1. The table on the left shows requests for the three pages A, B, and C at different times. An optimal schedule for this instance broadcasts pages B, C, A, B, C at times $1, 2, 3, 4, 5$ respectively. The table on the right of the figure shows the response time for each request in the optimal schedule. Adding up the response time of each request gives us the total response time of 25.

Input: r_t^p					
	t=0	t=1	t=2	t=3	t=4
page A	3	2	2	0	0
page B	2	0	2	0	0
page C	0	2	0	0	2

Response time: $r_t^p(S_t^p - t)$					
	t=0	t=1	t=2	t=3	t=4
page A	9	4	2	0	0
page B	2	0	4	0	0
page C	0	2	0	0	2

Fig. 1. The table on the left is an example input and the table on the right shows the response time for each request in an optimal schedule of broadcasting pages B, C, A, B, C at times $1, 2, 3, 4, 5$ respectively

3 Integer Programming Formulation

The Broadcast Scheduling Problem can be formulated as an integer program as follows. The binary variable $y_{t'}^p = 1$ iff page p is broadcast at time t'. The binary variable $x_{tt'}^p = 1$ iff a request (p, t) is satisfied at time $t' > t$ i.e. $y_{t'}^p = 1$ and $y_{t''}^p = 0, t < t'' < t'$. The constraints (2) ensure that whenever a request (p, t) is satisfied at time t', page p is broadcast at t'. Constraints (3) ensure that every request (p, t) is satisfied at some time $t' > t$. Constraints (4) ensure that at most one page is broadcast at any given time.

$$\min \sum_p \sum_t \sum_{t'=t+1}^{T+n} (t' - t) \cdot r_t^p \cdot x_{tt'}^p \tag{1}$$

subject to

$$y_{t'}^p - x_{tt'}^p \geq 0, \quad \forall p, t, t' > t \tag{2}$$

$$\sum_{t'=t+1}^{T+n} x_{tt'}^p \geq 1, \quad \forall p, t \tag{3}$$

$$\sum_p y_{t'}^p \leq 1, \quad \forall t' \tag{4}$$

$$x_{tt'}^p \in \{0, 1\}, \quad \forall p, t, t' \tag{5}$$

$$y_{t'}^p \in \{0, 1\}, \quad \forall p, t' \tag{6}$$

The corresponding linear programming (LP) relaxation can be obtained by letting the domain of $x_{tt'}^p$ and $y_{t'}^p$ be $0 \leq x_{tt'}^p, y_{t'}^p \leq 1$. For the example in Fig. 1, running an LP solver produces a fractional schedule that broadcasts pages $\{A,B\},\{A,C\},\{A,B\},\{A,B\},\{C\}$ at times $1,2,3,4,5$ respectively, where broadcasting $\{P_1,P_2\}$ at any time t means that exactly half of page P_1 and half of page P_2 are broadcast at time t. The cost of this fractional solution is 24.5.

4 Outline of the Algorithm

Let I be the given instance of the problem. The algorithm solves the LP for I to obtain an optimal (fractional) solution. It uses the LP solution to create a simplified instance I'. A $\frac{1}{\alpha}$-speed ($\frac{1}{\alpha}$ is an integer) fractional solution is constructed for instance I', which is converted into a $\frac{1}{\alpha}$-speed integral solution for I' using a min-cost flow computation. The integral solution for I' is then converted into a schedule for I.

5 Algorithm

Step I: Let I be the given instance of the problem. For any page p, let $N_p = \{t_1, t_2, \dots, t_{f_p}\}$ denote the times at which requests for page p are made in instance I. We first solve the LP for I to obtain an optimal fractional solution (x,y). Let $ft(\alpha,p,t)$ be the first time instance when α-fraction of request (p,t) get satisfied in the LP solution, i.e. $ft(\alpha,p,t) = \min\{t'' | \sum_{t'=t+1}^{t''} x^p_{tt'} \geq \alpha\}$. We consolidate the requests in I, transforming the instance I into a simplified instance I' which has the following property. If N'_p represents the times of positive requests for page p in I' then for any times $\{t'_u, t'_v\} \subseteq N'_p$, such that $t'_u < t'_v$, we have $ft(\alpha,p,t'_u) \leq t'_v$, where α is any fixed fraction in $(0,\frac{1}{2}]$. For every request (p,t) that is grouped with a request $(p,g(p,t)), g(p,t) \geq t$, we have $ft(\alpha,p,t) > g(p,t)$. Let $r'^p_{g(p,t)}$ denote the number of requests $(p,g(p,t))$ in I'. This transformation is done separately for each page. The pseudo-code for this transformation is given in Fig. 2.
Step II: Now we find a $\frac{1}{\alpha}$-speed fractional solution to instance I'. Let $N'_p = \{t'_1, t'_2, \dots, t'_{f_p}\}$ be the times of positive requests for page p in I'. Note that at least α-fraction of each request $(p,t'_i) \in I'$ gets satisfied before t'_{i+1} in the LP solution (x,y), i.e. $ft(\alpha,p,t'_i) \leq t'_{i+1}$. Consider the solution (x_α, y), where

$$x^p_{\alpha tt'} = \begin{cases} x^p_{tt'} & \text{if } t' < ft(\alpha,p,t) \\ \alpha - \sum_{t''=t+1}^{t'-1} x^p_{tt''} & \text{if } t' = ft(\alpha,p,t) \\ 0 & \text{otherwise} \end{cases}$$

By scaling all the y values and the x_α values by $\frac{1}{\alpha}$ we obtain a feasible $\frac{1}{\alpha}$-speed fractional solution, $(\frac{1}{\alpha}x_\alpha, \frac{1}{\alpha}y)$.
Step III: To obtain a $\frac{1}{\alpha}$-speed integer solution to instance I', we construct a minimum cost flow network N. N is the flow network that consists of a bipartite graph $G = (X,Y,E)$. Each node in X corresponds to a request $(p,t') \in I'$. Each node in Y corresponds to a time instance at which a page can be broadcast. There is an edge between $(p,t) \in X$ and $t' \in Y$ if $x_{\alpha tt'} > 0$. Note that for any

Request Consolidation(page p, α)

$N'_p \leftarrow \{t_{f_p}\}$
$l \leftarrow f_p$
$r'^p_{t_l} \leftarrow r^p_{t_l}$
for $k \leftarrow l-1$ **down to** 1 **do**
    $ft(\alpha, p, t_k) \leftarrow \min_{t'} \{\sum_{t=t_k+1}^{t'} x^p_{t_k t} \geq \alpha\}$
    if $(ft(\alpha, p, t_k) \leq t_l)$ **then**
        $N'_p \leftarrow N'_p \cup \{t_k\}$
        $l \leftarrow k$
        $r'^p_{t_l} \leftarrow r^p_{t_l}$
    else
        $g(p, t_k) \leftarrow t_l$
        $r'^p_{t_l} \leftarrow r'^p_{t_l} + r^p_{t_k}$
        $r'^p_{t_k} \leftarrow 0$
return N'_p

Fig. 2. Algorithm for Consolidating Requests

two nodes in X, say (p_1, t'_1) and (p_2, t'_2), that share a neighbor in Y, $p_1 \neq p_2$. This edge has a lower capacity of 0 and an upper capacity of 1 and a cost of $r'^p_t(t'-t)$. In addition, we have a source and a sink in N. There is an edge of 0 cost from the source to a node $(p, t) \in X$ with a lower and upper capacity of 1. Each edge from a node $t \in Y$ to the sink is of 0 cost and has a lower capacity of 0 and an upper capacity of $\frac{1}{\alpha}$. Figure 3 illustrates the min-cost flow network N. A $\frac{1}{\alpha}$-speed integral schedule can be obtained by setting $y^p_{t'} = f((p,t), t')$, where $f(a, b)$ denotes the flow along edge $(a, b) \in N$.

6 Analysis

Lemma 1. *In Step I, recall that each request $(p,t) \in I$ is grouped with a request $(p, g(p,t))$ to form instance I', where $g(p,t) \geq t$. If S and S' are integral solutions to instance I and I' respectively, then*

$$cost(S) \leq cost(S') + \sum_{(p,t) \in I} r^p_t(g(p,t) - t) \ .$$

Proof. The inequality follows easily because the additional cost of converting a solution for request $(p, g(p,t)) \in I'$ to a solution for request $(p,t) \in I$ is exactly $r^p_t(g(p,t)-t)$. Summing this over all requests will give us the required inequality.

Recall that (x, y) is an optimal LP solution for I. Define $C_{(p,t)}$ to be the cost of satisfying one request $(p,t) \in I$ in the LP solution (x,y), i.e. $C_{(p,t)} = \sum_{t'=t+1}^{T+n} (t'-t) x^p_{tt'}$. For any fixed fraction $\alpha \in (0, \frac{1}{2}]$, let $\beta = 1 - \alpha$. For any request $(p, g(p,t)) \in I'$, define $C'^{\theta}_{(p,g(p,t))}$ as the cost of satisfying exactly θ-fraction of request $(p, g(p,t)) \in I'$, i.e. $C'^{\theta}_{(p,g(p,t))} = \sum_{t'=g(p,t)+1}^{T+n} (t' - g(p,t)) x^p_{\theta g(p,t) t'}$.

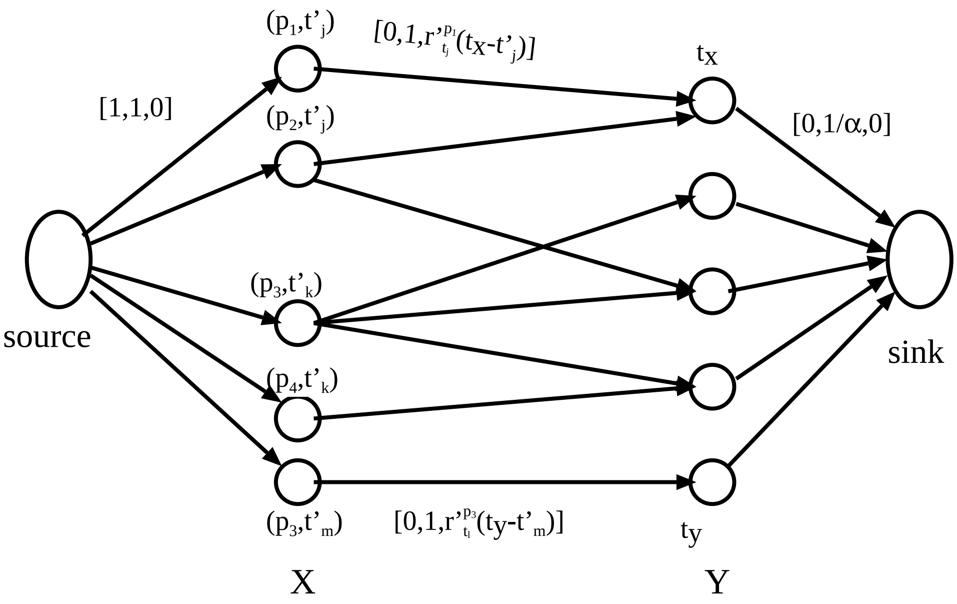

Fig. 3. The minimum cost flow network N. Each edge $e \in N$ has a label of the form $[l, u, c]$, where l and u denote the lower and the upper bounds respectively on the amount of flow through e and c denotes the cost per unit flow

Note that the cost of the $\frac{1}{\alpha}$-speed fractional solution for instance I' is given by $\frac{1}{\alpha}\sum_{(p,g(p,t))\in I'} r'^{p}_{g(p,t)} C'^{\alpha}_{(p,g(p,t))} = \frac{1}{\alpha}\sum_{(p,t)\in I} r^p_t C'^{\alpha}_{(p,g(p,t))}$ and the cost of the optimal fractional solution for instance I is given by $\sum_{(p,t)\in I} r^p_t C_{(p,t)}$. Lemma 2 relates the two costs.

Lemma 2. *The cost of $\frac{1}{\alpha}$-speed fractional solution in I' and the cost of an optimal fractional solution for instance I are related as follows.*

$$\frac{1}{\alpha}\sum_{(p,t)\in I} r^p_t C'^{\alpha}_{(p,g(p,t))} \leq \frac{1}{\beta}\sum_{(p,t)\in I} r^p_t C_{(p,t)} - \sum_{(p,t)\in I} r^p_t (g(p,t)-t) \ .$$

Proof. For the case when $t = g(p,t)$, the inequality follows because $(g(p,t)-t) = 0$ and $C_{(p,t)}$ is the cost of satisfying a complete request whereas $C'^{\alpha}_{(p,g(p,t))}$ is the cost of satisfying α-fraction of the request. For the case when $t < g(p,t)$, we know that $ft(\alpha,p,t) > g(p,t)$. This means that at least $\beta = (1-\alpha)$-fraction of the request $(p,t) \in I$ gets satisfied after $g(p,t)$. In other words, broadcast of page p between $g(p,t)$ and $ft(\beta,p,g(p,t))$ partially satisfies request (p,t) and $(p,g(p,t))$ in I. Thus we have

$$C'^{\beta}_{(p,g(p,t))} + \beta(g(p,t)-t) \leq C_{(p,t)} \ . \tag{7}$$

Multiplying (7) by $\frac{1}{\beta}$ and rearranging the terms gives us

$$\frac{1}{\beta}C'^{\beta}_{(p,g(p,t))} \leq \frac{1}{\beta}C_{(p,t)} - (g(p,t) - t) \ . \tag{8}$$

The left side of (8) represents the average cost of satisfying β-fraction of request $(p, g(p,t)) \in I'$. The cost of satisfying exactly α-fraction of the request, $C'^{\alpha}_{(p,g(p,t))}$, is upper bounded by $\frac{\alpha}{\beta}C'^{\beta}_{(p,g(p,t))}$, which is obtained by multiplying (8) by α, thus giving us

$$C'^{\alpha}_{(p,g(p,t))} \leq \frac{\alpha}{\beta}C'^{\beta}_{(p,g(p,t))} \leq \frac{\alpha}{\beta}C_{(p,t)} - \alpha(g(p,t) - t) \ . \tag{9}$$

Multiplying (9) by $\frac{1}{\alpha}r^p_t$ and summing over all requests gives us the cost of $\frac{1}{\alpha}$-speed fractional solution as required.

Lemma 3. *For any feasible flow in the minimum cost flow network, N, there is a $\frac{1}{\alpha}$-speed feasible fractional solution for instance I' of the same cost.*

Proof. Let $f(a,b)$ denote the flow along edge $(a,b) \in N$. Set $x^p_{\alpha tt'} = \alpha f((p,t),t')$ and $y^p_{t'} = x^p_{\alpha tt'}$. Since we know that for any node $(p,t) \in X$, $\sum_{t'} f((p,t),t') = 1$, we have $\frac{1}{\alpha}\sum_{t'} x^p_{\alpha tt'} = 1$. We can show that we have a $\frac{1}{\alpha}$-speed solution as follows. We know that $\sum_{(p,t)} f((p,t),t') \leq \frac{1}{\alpha}$, which implies that $\frac{1}{\alpha}\sum_{(p,t)} x^p_{\alpha tt'} \leq \frac{1}{\alpha}$ and hence $\frac{1}{\alpha}\sum_p y^p_{t'} \leq \frac{1}{\alpha}$. Thus feasibility is ensured. The cost of the $\frac{1}{\alpha}$-speed solution equals

$$\frac{1}{\alpha}\sum_{(p,t)\in I'}\sum_{t'>t} r^p_t(t'-t)x^p_{\alpha tt'} = \sum_{(p,t)\in X}\sum_{t'\in Y} r^p_t(t'-t)f((p,t),t') \ .$$

This proves that the $\frac{1}{\alpha}$-speed solution has the same cost as the flow in N.

Lemma 4. *There is a flow in N of the same cost as the $\frac{1}{\alpha}$-speed feasible fractional solution.*

Proof. Let the flow along an edge $((p,t),t')$ equal $\frac{1}{\alpha}x^p_{\alpha tt'}$. By definition of x_α, the capacity constraint of each edge is satisfied. Since we have a $\frac{1}{\alpha}$-speed *feasible* solution, $\frac{1}{\alpha}\sum_{t'} x^p_{\alpha tt'} = 1$ and hence the flow is conserved at every node in X which has an incoming flow of 1 unit. Since our solution is $\frac{1}{\alpha}$-speed, $\frac{1}{\alpha}\sum_p y^p_{t'} \leq \frac{1}{\alpha}$. Set flow along any edge from node t' to the sink equal to $\frac{1}{\alpha}\sum_p y^p_{t'}$. Note that the flow is also conserved at a node $t' \in Y$ as we know that $y^p_{t'} \geq x^p_{\alpha tt'}$. The cost of the solution equals

$$\sum_{(p,t)\in X}\sum_{t'\in Y} r^p_t(t'-t)f((p,t),t') = \frac{1}{\alpha}\sum_{(p,t)\in I'}\sum_{t'} r^p_t(t'-t)x^p_{\alpha tt'} \ .$$

Thus the cost of the flow in N is the same as the cost of the $\frac{1}{\alpha}$-speed solution.

Lemma 5. *There exists a $\frac{1}{\alpha}$-speed integral solution for I' of the same cost as the $\frac{1}{\alpha}$-speed fractional solution, $(\frac{1}{\alpha}x_\alpha, \frac{1}{\alpha}y)$.*

Proof. From Lemma 3 and Lemma 4 we know that the minimum cost flow in N equals the cost of the $\frac{1}{\alpha}$-speed fractional solution $(\frac{1}{\alpha}x_\alpha, \frac{1}{\alpha}y)$. By the integrality theorem, we can determine in polynomial time a minimum cost integral flow f^* in N that satisfies the capacity constraints. Using this integral flow f^* we can derive a $\frac{1}{\alpha}$-speed integral solution for instance I' using Lemma 3.

Theorem 6. *There is a $\frac{1}{\alpha}$-speed, $\frac{1}{1-\alpha}$-approximation solution for the Broadcast Scheduling Problem.*

Proof. We will prove that the algorithm in Sect. 5 gives a $\frac{1}{\alpha}$-speed, $\frac{1}{1-\alpha}$-approximation solution. From Lemma 2 and Lemma 5, we know that the cost of the $\frac{1}{\alpha}$-speed integral solution equals to

$$\frac{1}{\alpha}\sum_{(p,t)\in I} r_t^p C'^{\alpha}_{(p,g(p,t))} \le \frac{1}{\beta}\sum_{(p,t)\in I} r_t^p C_{(p,t)} - \sum_{(p,t)\in I} r_t^p (g(p,t)-t) \ .$$

Substituting this expression for $cost(S')$ in Lemma 1, we get

$$cost(S) \le \frac{1}{\beta}\sum_{(p,t)\in I} r_t^p C_{(p,t)} \le \frac{1}{\beta} OPT = \frac{1}{1-\alpha} OPT \ .$$

Corollary 7. *There is a* 2*-speed,* 2*-approximation solution and a* 3*-speed,* 1.5*-approximation solution for the Broadcast Scheduling Problem.*

Proof. The proof follows easily from Theorem 6 by setting $\alpha = \frac{1}{2}$ and $\alpha = \frac{1}{3}$ respectively for a 2-speed and a 3-speed solution.

7 4-Speed 1-Approximation Algorithm

Erlebach and Hall [14] showed that one can use a 6-speed algorithm to get a 1-approximation. This is done by taking a fractional solution obtained by solving a linear program, and then applying randomized rounding to generate schedules for 2 channels. On the remaining 4 channels they use the 4-speed algorithm by [18]. They are able to prove that the expected cost of each request is upper bounded by its fractional cost.

In this section, we show that one can obtain a random *fractional* schedule using the approach outlined in our paper. We can establish an upper bound on the cost of this fractional schedule. Finally, we convert this fractional schedule to an integral schedule using the network flow approach.

We obtain a 4-speed fractional solution as follows. On two channels, we select two pages (independently) with probability y_t^p at each time t. For the remaining two channels, we take our 2-speed *fractional* solution constructed in step II of section 5. Note that in this 2-speed fractional solution, a request (p,t) is satisfied by the same broadcasting as $(p, g(p,t))$ because we construct a 2-speed fractional solution after merging (p,t) to $(p, g(p,t))$. Therefore the cost of satisfying one request (p,t) in the 2-speed fractional solution is $2C'^{1/2}_{(p,g(p,t))} + (g(p,t)-t)$.

Lemma 8. *The expected cost of satisfying a request (p, t) by the 4-speed fractional schedule is at most the optimal LP cost for a request (p, t).*

Proof. In our 4-speed fractional schedule, each request (p, t) could be satisfied by either of the two random channels and *if* it is not satisfied by time $g(p, t)$, the 2-speed fractional schedule will satisfy it fractionally. Thus the expected cost of satisfying a request (p, t) by the 4-speed fractional schedule is at most

$$A = \sum_{t'=t+1}^{g(p,t)} \left(\left(\prod_{t''=t+1}^{t'-1} (1 - y^p_{t''})^2\right)(1 - (1 - y^p_{t'})^2)(t' - t)\right) + \left(\prod_{t'=t+1}^{g(p,t)} (1 - y^p_{t'})^2\right)(t^* - t),$$

where $(t^* - t) = 2C'^{1/2}_{(p,g(p,t))} + (g(p, t) - t)$. The optimal LP cost for a request (p, t) is at least

$$B = \sum_{t'=t+1}^{g(p,t)} y^p_{t'} \cdot (t' - t) + (1 - \sum_{t'=t+1}^{g(p,t)} y^p_{t'}) \cdot (t^* - t) \ .$$

Because $\sum_{t'=t+1}^{g(p,t)} y^p_{t'} < 1/2$ and $t^* > g(p, t)$, we can get $A \leq B$ using the same proof as [14].

Lemma 9. *We can find a 4-speed fractional schedule of the cost at most OPT in polynomial time.*

Proof. By Lemma 8 and the linearity of expectation, the expected cost of this 4-speed fractional schedule is at most OPT. Using standard derandomization techniques, we can obtain a deterministic polynomial-time algorithm that yields a 4-speed fractional schedule of the cost at most OPT.

Now we convert the 4-speed fractional schedule to a 4-speed integral schedule. Because the schedule on two random channels is already integral, we only need to convert the 2-speed fractional schedule to an integral schedule without losing any cost.

Lemma 10. *We can convert the 4-speed fractional schedule to a 4-speed integral schedule of the same cost.*

Proof. By constructing the minimum cost flow network only for the requests satisfied by the fractional schedule, we can convert the 2-speed fractional solution to 2-speed integral solution of the same cost. Combining with two random channels, we have a 4-speed integral solution.

Theorem 11. *There is a 4-speed, 1-approximation solution for the Broadcast Scheduling Problem.*

8 Experiments

Although we cannot obtain 1-speed algorithm which has an $O(1)$ approximation guarantee, we devised several simple heuristics and compared their performance with LP and IP solutions. The results showed that we can get very close to the optimal solution for randomly generated instances.

We have two types of heuristics. The first heuristics use the number of outstanding requests and the time of next request. In the second heuristics, we first solve LP and use the solution to guide the construction of our schedule.

8.1 Heuristic 1

Let N_t^p be the number of requests for page p which are not yet satisfied at the end of timeslot t. We may want to broadcast a page with the largest N_t^p. Another useful information to consider is the next request time. Let C_t^p be the next request time for page p after time t. If C_t^p is small, it may be better to delay broadcasting the page till the next request time, by which a single broadcast satisfies them with a small delay. We tried several heuristics using combinations of these two values.

One simple heuristic is to compute N_t^p (C_t^p) for all pages p at each time t and broadcast a page j which satisfies $N_t^j \geq N_t^k$ ($C_t^j \geq C_t^k$) for any page k. These heuristics which consider one metric (N_t^p or C_t^p) did not perform very well because the other value can be too small.

To reflect both values at the same time, a reasonable metric is $N_t^p \cdot C_t^p$. In addition, if N_t^p is very small, then we may ignore the page even if its C_t^p value is large enough to make $N_t^p \cdot C_t^p$ the largest. At each time t, we compute both N_t^p and C_t^p and broadcast a page j which satisfies $N_t^j \cdot C_t^j \geq N_t^k \cdot C_t^k$ for any page k of which N_t^k is within top $\alpha\%$. We varied α from 20% to 100% in the experiments.

8.2 Heuristic 2

In this type of heuristic, we solve LP to get an optimal fractional solution (x, y) and compute z_t^p as follows.

$$z_t^p = \begin{cases} z_{t-1}^p + y_t^p & \text{if } N_t^p > 0 \\ 0 & \text{otherwise} \end{cases}$$

z_t^p can be interpreted as preference to broadcasting page p at time t. Thus we choose a page with the largest z_t^p value at each time. We call this heuristic as *Deterministic LP Rounding.*

A variation of this heuristic is to use randomization. We normalize z_t^p by dividing it by $\sum_p z_t^p$ so that z_t^p represents the probability to broadcast page p at time t. Then we choose a page randomly based on z_t^p distribution. We call this heuristic as *Randomized LP Rounding.*

8.3 Settings

We experimented these heuristics with two types of inputs.

- Uniformly random input generator: Over all possible time and possible pages, there are approximately $(density) \cdot n \cdot T$ uniformly distributed nonzero demands. If a demand is nonzero, it is a random number from 1 to $(maximum\ demand)$.
- Zipf Distribution input generator: At every time, the total number of requests, summing over all pages, is a random number from 1 to $(density) \cdot n \cdot (maximum\ demand)$. Each request has probability of $\frac{1}{i \cdot H(n)}$ of requesting page i, where $H(n) = \sum_{i=1}^{n} 1/i$. In other words, the choice of pages follows Zipf distribution [24] with $\theta = 0$. Thus, page 1 has most requests and page n has least. This corresponds to the *measurements* performed in [12] (for a movies-on-demand application).

In our experiment, the uniformly random input consists of 10 pages ($n = 10$). The time of last request is 50 ($T = 50$). The *density* of demand distribution in the uniformly random input generator is 0.4. The *maximum demand* of a request is 20.

Similar to the uniformly random input generator, the Zipf distribution input consists of 10 pages, and the time of last request is 50. The maximum total number of requests, summing over all pages, is $0.4 \cdot 10 \cdot 20 = 80$, so that the expected total demand is the same as that in the uniformly random input.

To get the expected result of the Randomized LP Rounding heuristic, the result of this heuristic is the averaged cost over 100 runs. *Randomized LP Rounding (best over 100 run)* is the lowest cost over 100 randomized LP rounding.

We ran our experiments on a Sun Ultra-5 workstation, using ILOG CPLEX 6.5.2 to solve LP and IP, and Matlab 5.3 to implement all heuristics.

8.4 Results

Table 1 and 2 shows the ratio of our heuristic solutions to IP optimal solution. We repeated the experiment 150 times for uniformly random input, and another 150 times for Zipf distribution input and took averages. The Zipf input seems harder to solve. It takes 1000 to 4000 seconds to find an optimal solution, while it takes less than 100 seconds to find a LP solution. The uniformly random input problems are easier to solve. It takes less than 20 seconds to find an optimal solution for most problems. To find a LP solution, it takes a few seconds.

To find a solution using first set of heuristics, it takes around 0.1 seconds. LP rounding takes around 0.1 seconds too.

Deterministic LP rounding method performed best for both uniform and Zipf inputs. The difference was only about 0.9% for uniform input and 1.6% for Zipf input. Moreover, in a few cases, deterministic LP rounding can round a non optimal solution to an optimal solution, as shown from its "% same as OPT" is higher than that of LP.

Table 1. Percentage change relative to OPT for uniformly distributed input

Different Heuristics	Mean	Median	Min	Max	S.D.	% same as OPT
LP	-0.02	0.00	-0.37	0.00	0.06	68.00
N_i	25.76	25.27	10.01	41.40	6.13	0.00
C_i	60.13	57.86	34.26	116.70	16.32	0.00
$N_i \cdot C_i$	9.57	9.08	3.15	19.61	3.38	0.00
$N_i \cdot C_i$ (top 20%)	12.96	12.60	4.50	27.33	3.99	0.00
$N_i \cdot C_i$ (top 50%)	9.31	8.83	3.15	19.21	3.38	0.00
$N_i \cdot C_i$ (top 80%)	9.57	9.08	3.15	19.61	3.38	0.00
Deterministic LP Rounding	0.90	0.00	0.00	11.47	2.21	68.67
Randomized LP Rounding	7.17	0.00	0.00	39.95	11.61	67.33
Ditto (best over 100 runs)	2.15	0.00	0.00	18.45	4.67	67.33

Table 2. Percentage change relative to OPT for Zipf distribution input

Different Heuristics	Mean	Median	Min	Max	S.D.	% same as OPT
LP	-0.08	-0.07	-0.31	0.00	0.08	20.00
N_i	17.82	17.35	13.21	24.54	2.48	0.00
C_i	120.80	109.20	62.86	195.20	34.82	0.00
$N_i \cdot C_i$	12.81	12.71	8.12	18.18	1.64	0.00
$N_i \cdot C_i$ (top 20%)	15.25	14.97	9.59	19.79	2.49	0.00
$N_i \cdot C_i$ (top 50%)	12.65	12.65	6.25	16.68	1.98	0.00
$N_i \cdot C_i$ (top 80%)	12.61	12.36	8.12	16.55	1.72	0.00
Deterministic LP Rounding	1.54	1.47	0.00	5.29	1.34	22.00
Randomized LP Rounding	19.18	24.63	0.00	30.21	10.41	20.00
Ditto (best over 100 runs)	10.33	12.94	0.00	19.08	6.19	20.00

As for the first type of heuristics, $N_i \cdot C_i$ metric performed best. Although it does not perform as good as deterministic LP rounding, it runs much faster since it needs not to solve a LP. In this method, we only considered the pages which have large N_i and varied the percentage as 20%, 50%, 80%, 100%. 50% performed best for the uniform input and 80% for the Zipf input. If the heuristic considers only the requests with N_i larger than the highest 20-percentile among all N_i, it cuts too many good candidates. If the heuristic simply ranks all requests by $N_i \cdot C_i$, it may broadcast a page with small N_i but very large C_i. However, it is better delay this page and broadcast a page with larger N_i.

References

1. S. Acharya. "Broadcast Disks": Dissemination-based data management for asymmetric communication environments. Ph.D. Thesis, Brown University, 1998.
2. S. Acharya, M. Franklin, and S. Zdonik. Dissemination-based data delivery using broadcast disks. In *IEEE Personal Communications*, 2(6), 1995.
3. S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast Disks: Data management for asymmetric communications Environments. In *Proc. of ACM SIGMOD International Conference on Management of Data* (SIGMOD), 199-210, 1995.
4. D. Aksoy. On-demand data broadcast for large-scale and dynamic applications. Ph.D. Thesis, University of Maryland at College Park, 2000.
5. D. Aksoy, and M. Franklin. RxW: A scheduling approach for large-scale on-demand data broadcast. In *IEEE/ACM Transactions On Networking*, Volume 7, Number 6, 486-860, 1999.
6. M. H. Ammar and J. W. Wong. The design of teletext broadcast cycles. In *Performance Evaluation*, Vol. 5(4), 235-242, 1985.
7. A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. In *Proceedings of 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 11-20, 1998.
8. Y. Bartal and S. Muthukrishnan. Minimizing maximum response time in scheduling broadcasts. In *Proceedings of 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, 558-559, 2000.
9. R. Bhatia. Approximation algorithms for scheduling problems. Ph.D. Thesis, University of Maryland at College Park, 1998.
10. M. Charikar, S. Guha, É. Tardos, and D. Shmoys. A constant factor approximation for the k-median problem. In *Proc. of 31st Annual ACM Symposium on Theory of Computing*, 1-10, 1999.
11. C. Chekuri, S. Khanna and A. Zhu. Algorithms for minimizing weighted flow time. In *Proc. of 33rd Annual ACM Symp. on Theory of Computing*, 84-93, 2001.
12. A. L. Chervenak. Tertiary Storage: An Evaluation of New Applications. *Ph.D. Thesis, UC Berkeley*, 1994.
13. DirecPC website, http://www.direcpc.com
14. T. Erlebach, A. Hall. NP-Hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. In *Proc. of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 194-202, 2002.
15. Intel intercast website, http://www.intercast.com

16. T. S. Jayram, T. Kimbrel, R. Krauthgamer, B. Schieber, and M. Sviridenko. Online server allocation in a server farm via benefit task systems. In *Proc. of 33rd Annual ACM Symp. on Theory of Computing*, 540-549, 2001.
17. C. Kenyon, N. Schabanel, and N. Young. Polynomial-time approximation scheme for data broadcast. In *Proc. of 32nd Annual ACM Symposium on Theory of Computing* 659-666, 2000.
18. B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. In *European Symposium of Algorithms*, LNCS 1879, Springer-Verlag, 290-301, 2000.
19. B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. In *IEEE Symposium on Foundations of Computation*, 214-221, 1995.
20. C. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *Proc. of 29th Annual ACM Symposium on Theory of Computing*, 140-149, 1997.
21. M. S. Squillante, D. D. Yao and L. Zhang. Web traffic modelling and web server performance analysis. In *Proc. of 38th IEEE Conf. on Decision and Control*, 4432-4437, 1999.
22. H. Kaplan, R. E. Tarjan, and K. Tsioutsiouliklis. Faster kinetic heaps and their use in broadcast scheduling. In *Proc. of 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 836-844, 2001.
23. J. Wong. Broadcast Delivery. In *Proc. of the IEEE*, 76(12), 1988.
24. D. E. Knuth. The Art of Computer Programming, Volume 3. Addison-Wesley, 1973.