Off-line Algorithms for Minimizing Total Flow Time in Broadcast Scheduling

Wun-Tat Chan[1⋆], Francis Y. L. Chin[1⋆⋆], Yong Zhang[1], Hong Zhu[2], Hong Shen[3], and Prudence W.H. Wong[4⋆⋆⋆]

[1] Department of Computer Science, University of Hong Kong, Hong Kong
{wtchan, chin, yzhang}@cs.hku.hk
[2] Department of Computer Science and Engineering, Fudan University, China
hzhu@fudan.edu.cn
[3] Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Japan, shen@jaist.ac.jp
[4] Department of Computer Science, University of Liverpool, UK
pwong@csc.liv.ac.uk

Abstract. We study the off-line broadcast scheduling problem to minimize total (or average) flow time. Assume the server has k pages and the requests arrive at n distinct times, we give the first algorithm to find the optimal schedule for the server with a single channel, in $O(k^3(n+k)^{k-1})$ time. For m-channel case, i.e., the server can broadcast m different pages at a time where $m < k$, we find the optimal schedule in $O(n^{k-m})$ time when k and m are constant. In the single channel case, we also give a simple linear-time approximation algorithm to minimize average flow time, which achieves an additive $(k-1)/2$-approximation.

1 Introduction

In an on-demand broadcasting system, the server receives requests for pages from clients over time, and answers these requests by broadcasting (sending) the pages via the broadcast channels. After the server broadcasts a page, all pending requests for that page are satisfied. The scheduler of the server is to arrange the order of the page broadcasts so as to minimize the total (or average) delay of the requests. In this paper we assume that time is discrete, all the pages have unit length and the server has m broadcast channels, i.e., at most m different pages can be broadcast at each time slot. We formalize the problem as follows. Assume that the server contains k pages, namely $P_0, P_1, \ldots, P_{k-1}$, requested by clients at integral time only. Let r_{ti} denote the number of requests for P_i at time t. For a schedule, let b_{ti} be the earliest time at or after time t when P_i is broadcast. The *flow time* of a request for P_i at time t is thus $b_{ti} - t + 1$. Suppose that the last requests arrive at time n. The total flow time of the schedule, which is to be minimized, is equal to $\sum_{t=0}^{n} \sum_{i=0}^{k-1} r_{ti}(b_{ti} - t + 1)$. Note that an optimal schedule that minimizes the total flow time is also an optimal schedule that minimizes the

⋆ This research was supported in part by Hong Kong RGC Grant HKU-5172/03E.
⋆⋆ This research was supported in part by Hong Kong RGC Grant HKU-7142/03E.
⋆⋆⋆ This research was supported in part by Nuffield Foundation Grant NAL/01004/G.

average flow time. In this paper we consider the off-line version of the problem, in which the server is aware of all the requests in advance.

Previous work of the problem considered that the number of broadcast channels $m = 1$ and the number of pages k is a variable. This problem was shown to be NP-hard by Erlebach and Hall [5]. Recently, Bansal et al. [1] gave an additive $O(\sqrt{k})$-approximation algorithm in minimizing the average flow time, yet their algorithm requires to solve a time-consuming linear programming. Besides, most of the previous works considered the resource augmentation setting. In the setting, an m-speed algorithm refers to an algorithm of a server with m broadcast channels and an m-speed c-approximation algorithm is an m-speed algorithm which produces the schedules with total (or average) flow time at most c times that of the schedule produced by the optimal 1-speed algorithm. Kalyanasundaram et al. [9] gave a $\frac{1}{\epsilon}$-speed $\frac{1}{1-2\epsilon}$-approximation algorithm for any fixed $\epsilon \in (0, \frac{1}{3}]$. Gandhi et al. [7] gave a $\frac{1}{\epsilon}$-speed $\frac{1}{1-\epsilon}$-approximation algorithm for any fixed $\epsilon \in (0, \frac{1}{2}]$. To match the performance of the 1-speed optimal algorithm, Erlebach and Hall [5] gave a 6-speed algorithm, which was improved to 4-speed [7] and then to 3-speed by Gandhi et al. [8]. The on-line version of the problem has been studied by Edmonds and Pruhs [3, 4]. Bartal and Muthukrishnan [2] have considered the problem with another objective function which is to minimize the maximum flow time.

The main result of this paper is to give the first optimal algorithm for the broadcast scheduling problem to minimize total flow time when the number of pages k is fixed. Based on a dynamic programming technique and the concave property of the optimization function, our algorithm constructs the optimal schedule for the case when $m = 1$ in $O(k^3(n+k)^{k-1})$ time where the last requests arrive at time n. When k is a constant, the time complexity is $O(n^{k-1})$. We generalize this result in the m-channel case where a server has $2 \leq m < k$ broadcast channels. We show that in this case the optimal schedule can be found in $O((k^3 + \binom{k-1}{k-m})(n + \frac{k}{m})^{k-m})$ time, or $O(n^{k-m})$ time when k and m are constants. Note that the problem with multiple channels seems to be easier than the problem with single channel and the multi-channel problem is not NP-hard if $k - m$ is a constant. In addition, we also give a simple approximation result for the case of $m = 1$. Different from all previous works that need to solve the time-consuming linear programming, we give a simple linear-time algorithm to achieve a tight additive $\frac{k-1}{2}$-approximation on minimizing the average flow time, i.e., the average flow time of the schedules produced by our algorithm is at most that of any schedule plus $(k-1)/2$. Even though our approximation algorithm seems inferior than the $O(\sqrt{k})$-approximation algorithm [1], our algorithm is much simpler and shall have comparable performance as k is usually small in practice.

The rest of the paper is organized as follows. Section 2 presents the optimal algorithms. A linear-time optimal algorithm for $m = 1$ and $k = 2$ is given in Section 2.1. For general k, the optimal algorithms for the cases of $m = 1$ and general m are given in Sections 2.2 and 2.3, respectively. Section 3 presents the approximation algorithm for $m = 1$ and general k.

2 Optimal Algorithms

2.1 Broadcast Scheduling for Two Pages

Assume we have $m = 1$ broadcast channel, and $k = 2$ pages, and the last request arrives at time n. Let F denote the minimum total flow time in satisfying all requests. Given that P_i is broadcast at time t where $i = 0$ or 1, for $0 \le t \le n+1$, let $F_i(t)$ denote the minimum total flow time in satisfying all the requests made at or after time t, i.e., the r_{ab} requests for P_b at time a for $t \le a \le n$ and $b \in \{0, 1\}$. Note that by definition $F_i(n+1) = 0$ for $i = 0$ or 1 because there is no request after time n. Otherwise, for example, $F_0(0)$ is the minimum total flow time to satisfy all requests given that P_0 is broadcast at time 0. As only P_0 or P_1 can be broadcast at time 0, we can see that $F = \min\{F_0(0), F_1(0)\}$. In the following we define $F_i(t)$ recursively. For the base case where $t = n+1$,

$$F_0(n+1) = 0 \quad \text{and} \quad F_1(n+1) = 0.$$

In general, we consider $0 \le t \le n$. For $F_0(t)$, the optimal schedule must have P_0 broadcast at each time $t, t+1, \ldots, s-1$ for some $s \ge t+1$, and then P_1 broadcast at time s. Thus,

$$F_0(t) = \min_{t+1 \le s \le n+1} \{c_1(s,t) + F_1(s)\}$$

where $c_1(s,t) = \sum_{i=t}^{s-1} (r_{i0} + r_{i1}(s-i+1))$ is the total flow time in satisfying those requests arrived within time t to time $s-1$ inclusively. Similarly,

$$F_1(t) = \min_{t+1 \le s \le n+1} \{c_0(s,t) + F_0(s)\}$$

where $c_0(s,t) = \sum_{i=t}^{s-1} (r_{i0}(s-i+1) + r_{i1})$.

With $O(n)$-time preprocessing (see Lemmas 8 and 9 in the Appendix), functions $c_0(s,t)$ and $c_1(s,t)$ can be computed in constant time for any given s and t. Hence, the brute-force method to find F by computing all $F_i(t)$ for $0 \le i \le 1$ and $0 \le t \le n$ takes $O(n^2)$ time. However, we show that it can be done in linear time by using the algorithm of Galil and Park [6]. We say that a function $w()$ is *concave* if it satisfies the *quadrangle inequality*, i.e., $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$ for $a \le b \le c \le d$. Galil and Park have the following theorem.

Theorem 1 (Galil and Park [6]). *Given a concave function $w(i,j)$ for integer $0 \le i \le j \le n$ and $E(0)$, the recurrence $E(j) = \min_{0 \le i < j}\{D(i) + w(i,j)\}$ for $1 \le j \le n$ can be solved in $O(n)$ time, if $D(i)$ can be computed in constant time.*

We show that our recurrences can be transformed to that of Theorem 1, and thus our recurrences can also be solved in linear time. We give the details for the case of $F_0()$ and the case of $F_1()$ can be done similarly. Let $E(j) = F_0(n-j+1)$ for $0 \le j \le n+1$. We have the base case $E(0) = F(n+1) = 0$. Let $w(i,j) = c_1(n-i+1, n-j+1)$ for $0 \le i < j \le n+1$. We have the recurrence $E(j) = \min_{0 \le i < j}\{D(i) + w(i,j)\}$ for $1 \le j \le n+1$, where $D(i) = F_1(n-i+1)$. Given that the relevant values of $F_1()$ (resp. $F_0()$) are already known when $D(i)$ is needed, $D(i)$ can be obtained in constant time. Lemma 1 shows that function $w(i,j)$ satisfies the quadrangle inequality. Therefore, by Theorem 1, we can find the optimal schedule in linear time, as given in Theorem 2.

Lemma 1. *The function* $w(i,j) = c_1(n-i+1, n-j+1)$ *(resp.* $c_0(n-i+1, n-j+1)$*) for integer* $0 \le i < j \le n+1$ *satisfies the quadrangle inequality, i.e.,* $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$ *for integer* $a \le b \le c \le d$*.*

Proof. We consider the case that $w(i,j) = c_1(n-i+1, n-j+1)$ and the case of $w(i,j) = c_0(n-i+1, n-j+1)$ can be proved similarly. By definition, $w(i,j) = c_1(n-i+1, n-j+1) = \sum_{x=n-j+1}^{n-i}(r_{x0} + r_{x1}(n-i-x+2))$. We can see that $\sum_{x=n-c+1}^{n-a} r_{x0} + \sum_{x=n-d+1}^{n-b} r_{x0} = \sum_{x=n-d+1}^{n-a} r_{x0} + \sum_{x=n-c+1}^{n-b} r_{x0}$ and $\sum_{x=n-c+1}^{n-a} r_{x1}(n-a-x+2) + \sum_{x=n-d+1}^{n-b} r_{x1}(n-b-x+2) = \sum_{x=n-c+1}^{n-a} r_{x1}(n-a-x+2) + \sum_{x=n-d+1}^{n-c} r_{x1}(n-b-x+2) + \sum_{x=n-c+1}^{n-b} r_{x1}(n-b-x+2) \le \sum_{x=n-d+1}^{n-a} r_{x1}(n-a-x+2) + \sum_{x=n-c+1}^{n-b} r_{x1}(n-b-x+2)$. The last inequality is due to $n-a \ge n-b$. Therefore we have $w(a,c) + w(b,d) \le w(b,c) + w(a,d)$.

Theorem 2. *The minimum total flow time of the* 2*-page broadcast scheduling problem with requests arriving at integer time* 0 *to time* n *can be computed in* $O(n)$ *time.*

2.2 Broadcast Scheduling for k Pages

In this section we consider the problem with a single broadcast channel and k pages, for any fixed integer k. Assuming that the last requests arrive at time n, we formulate the problem as a dynamic programming problem which is a generalization of that in Section 2.1. Each sub-problem in the dynamic programming can be specified by a k-dimensional vector $v = (v_0, \ldots, v_{k-1})$ where $0 \le v_i \le n+k-1$ and $v_i \ne v_j$ if $i \ne j$. Let $v_{min} = \min_{0 \le i \le k-1}\{v_i\}$. The sub-problem corresponding to v is to find the minimum total flow time for satisfying all the requests between v_{min} and n, i.e., the r_{tj} requests for P_j at time t for $v_{min} \le t \le n$ and $0 \le j \le k-1$, with v_i being P_i's earliest broadcasting time. For example, when $k = 2$, $F_0(t)$ defined in Section 2.1 refers to the minimum total flow time over all sub-problems corresponding to the vectors $v = (t, t')$ with $t' > t$. For general k, there are $O((n+k)^k)$ possible k-dimensional vectors as well as sub-problems, the time complexity will be at least $O((n+k)^k)$. In the following, we shall modify the definition of the vectors corresponding to the sub-problems slightly so that better than $O((n+k)^k)$ time can be achieved.

The vector $v = (v_0, \ldots, v_{k-1})$ is similar to what is defined earlier except that one of the v_i's value is unspecified, which is represented as "$*$". Suppose that $v_\alpha = *$, it means that in the sub-problem corresponding to v, the earliest broadcasting time of P_α is not fixed, yet it cannot be earlier than that of all other pages. Let $v_{min} = \min_{1 \le j \le k-1 \ \& \ v_j \ne *}\{v_j\}$. The sub-problem corresponding to v, say with $v_\alpha = *$, is to find the minimum total flow time for satisfying all the request between v_{min} and n with v_i being P_i's earliest broadcasting time for $i \ne \alpha$. P_α's earliest broadcasting time can be any possible value between $v_{min}+1$ and $n+k-1$, i.e., some $\beta \in C_v$ with $C_v = \{t \mid t \ne v_j$ for all $v_j \ne *$ and $v_{min}+1 \le t \le n+k-1\}$.

Let $F(v)$ denote the minimum total flow time of the sub-problem corresponding to v. We define $F(v)$ recursively as follows. In the base case, we have

$v_{min} = n + 1$ which is the largest possible value for v_{min} as v needs to specify $k-1$ distinct values between v_{min} and $n+k-1$. By definition,

$$F(v) = 0 \quad \text{for all } v \text{ with } v_{min} = n+1$$

because there is no request after time n. In general, we consider $0 \leq v_{min} \leq n$ and assume that $v_\alpha = *$. Although v_α is unspecified, it can take the value $\beta \in C_v$ only, i.e., P_α can be broadcast the earliest at some time $\beta \in C_v$ only. We have to consider the $|C_v|$ different cases of assigning a value $\beta \in C_v$ to v_α. Therefore, $F(v)$ equals the minimum total flow time among the sub-problems corresponding to v with v_α assigned time β, for each $\beta \in C_v$. Similar to the 2-page case, for each of these sub-problems, if $v_x = v_{min}$, then the optimal schedule must have P_x broadcast at each time $v_{min}, v_{min}+1, \ldots, s-1$ where $s = \min\{\beta, \min_{v_j \neq v_{min} \;\&\; v_j \neq *}\{v_j\}\}$ is the earliest broadcasting time among the pages other than P_x. Note that there is no pending request for P_x immediately after time $s-1$. Thus, each of these sub-problems depends on one "smaller" sub-problem which is corresponding to another vector $u = (u_0, \ldots, u_{k-1})$ constructed from v as follows.

$$u_i = \begin{cases} * & \text{for } i \text{ where } v_i = v_{min}, \\ \beta & \text{for } i \text{ where } v_i = *, \\ v_i & \text{otherwise,} \end{cases}$$

for each $\beta \in C_v$. To compute $F(v)$, we consider the $|C_v|$ different "smaller" sub-problems, i.e.,

$$F(v) = \min_{\text{Sub-problems } u \text{ derived from } v} \{F(u) + c(u,v)\}$$

where $c(u,v) = \sum_{i=0}^{k-1} \sum_{t=v_{min}}^{u_{min}-1} f_{ti}(u,v)$ is the total flow time of the r_{ti} requests for P_i at time t for $v_{min} \leq t \leq u_{min}-1$ and $0 \leq i \leq k-1$ and $f_{ti}(u,v)$ is the total flow time of the particular r_{ti} requests for P_i at time t, i.e.,

$$f_{ti}(u,v) = \begin{cases} r_{ti} & \text{for } i \text{ where } v_i = v_{min}, \\ r_{ti}(u_i - t + 1) & \text{otherwise.} \end{cases}$$

We give an analysis on a brute-force implementation in solving the above recurrence of $F(v)$. Then we show a faster implementation by generalizing the approach used in Section 2.1. Lemma 2 implies that there are $k(n+k)!/(n+1)!$ different sub-problems corresponding to a k-dimensional vector. Similar to the case of $k=2$ in Section 2.1, with $O(kn)$-time preprocessing, $\sum_{t=v_{min}}^{u_{min}-1} f_{ti}(u,v)$ can be computed in constant time for any given i, v_{min} and $u_{min}-1$ (see Lemmas 8 and 9 in the Appendix) and thus $c(u,v)$ can be computed in $O(k)$ time. Since the number of sub-problems corresponding to u, derived from a given v, is $O(n)$, a particular $F(v)$ can be computed in $O(kn)$ time. Therefore, the brute-force method in finding F by computing all $F(v)$ of the subproblems corresponding to v takes $O(k^2 n(n+k)!/(n+1)!)$, or concisely, $O(k^2(n+k)^k)$ time.

Lemma 2. *There are $k(n+k)!/(n+1)!$ different k-dimensional vectors $v = (v_0, \ldots, v_{k-1})$ with $0 \leq v_j \leq n-k+1$ for all $0 \leq j \leq k-1$ except one v_j's value equals $*$ and $v_i \neq v_j$ for all $i \neq j$.*

Proof. Since each $v_j \neq *$ should have a distinct value between $v_{min}+1$ and $n+k-1$, there are at most $\binom{n+k}{k-1}$ ways of choosing $k-1$ distinct values between 0 and $n+k-1$. As these $k-1$ distinct values have $k!$ ways of assigning to the k positions, there are $k(n+k)!/(n+1)!$ different valid vectors of v.

Note that up to this stage, there is no gain in the time complexity and $O((n+k)^k)$ time is still needed. In order to improve the time complexity, concave property of the function $c(u, v)$ should be exploited as in Section 2.1. In the faster implementation, we group the sub-problems into k^2 groups. Two sub-problems corresponding to v and u, respectively, belong to the same group G_{xy}, for $0 \leq x, y \leq k-1$, if $v_{min} = v_x$ and $u_{min} = u_x$, and $v_y = u_y = *$, i.e., the two sub-problems have the earliest broadcasting time for the same page P_x and the unspecified broadcasting time same for another page P_y. We further divide the sub-problems in each group into sub-groups. Two sub-problems corresponding to v and v', respectively, of the same group G_{xy} belong to the same sub-group if except P_x and P_y all other pages among the two sub-problems have the same corresponding earliest broadcasting time, i.e., $v_j = v'_j$ for all j with $j \neq x$ and $j \neq y$. There are $O(n)$ sub-problems in each sub-group and there are $(n+k-1)!/(n+1)!$ sub-groups in each group, as shown in the following lemma.

Lemma 3. *There are $(n+k-1)!/(n+1)!$ sub-groups in each group.*

Proof. WLOG, consider a particular group G_{01}. The number of sub-groups in G_{01} is equal to the number of ways in choosing $k-2$ distinct values between 1 and $n+k-1$ for $v_2, v_3, \ldots, v_{k-1}$, i.e., $\binom{n+k-1}{k-2}$ ways. As these $k-2$ distinct values have $(k-2)!$ ways of assigning to the $k-2$ positions, there are $(n+k-1)!/(n+1)!$ sub-groups in G_{01}. Similarly, it applies to each of the other groups.

We can compute $F(v)$ for all vectors v corresponding to the sub-problems of the same sub-group in $O(k(n+k))$ time. It is done by transforming the recurrence in this section to that of Theorem 1 in Section 2.1. WLOG, we consider a sub-group in G_{01}. For all vectors v corresponding to the sub-problems in the sub-group, we have $v_0 = v_{min}$ and $v_1 = *$ and all other v_j fixed. For ease of explanation, we assume that $v_j \in [n+2, n+k-1]$ for all $2 \leq j \leq k-1$. (The assumption is not necessary in proving the correctness of our algorithm.) For $0 \leq t \leq n+1$, let $E(t) = F(v)$ where $v_{min} = n-t+1$. Then we have $E(0) = F(v) = 0$ because $v_{min} = n+1$. For $0 \leq s \leq n+1$, let $D(s) = F(u)$ where $u_1 = n-s+1$ and u is derived from v. For $0 \leq s < t \leq n+1$, let $w(s,t) = c(u, v)$, which is given as follows.

$$w(s,t) = \sum_{i=n-t+1}^{n-s} r_{i0} + \sum_{i=n-t+1}^{n-s} r_{i1}(n-s-i+2) + \sum_{j=2}^{k-1} \sum_{i=n-t+1}^{n-s} r_{ij}(v_j - i + 1)$$

It can be verified that the function $w(s,t)$ satisfies the quadrangle inequality as in Lemma 4. Hence, by Theorem 1, all $F(v)$ for sub-problems corresponding to v of the same sub-group can be computed in $O(kn)$ time and all $F(v)$ for all sub-problems of "all" sub-groups can be computed in $O(kn \cdot k^2 \cdot (n+k-1)!/(n+1)!)$, or concisely, $O(k^3(n+k)^{k-1})$. Thus, we have Theorem 3.

Lemma 4. *The function $w(s,t)$ for $0 \leq s < t \leq n+1$ satisfies the quadrangle inequality, i.e., $w(a,c)+w(b,d) \leq w(b,c)+w(a,d)$ for $a \leq b \leq c \leq d$.*

Theorem 3. *The minimum total flow time of the k-page broadcast scheduling problem with requests arriving at integral times 0 to n can be computed in $O(k^3(n+k)^{k-1})$ time.*

2.3 Broadcast Scheduling with Multiple Channels

Assume that there are m broadcast channels available to the server. At each time slot, a server can broadcast at most m different pages among the k pages, where $m < k$. WLOG we can assume that there is an optimal schedule that broadcasts exactly m different pages at each time slot.

We apply the framework in solving the dynamic programming problem in Section 2.2 to this problem. Each sub-problem in the dynamic programming can be specified by a k-dimensional vector $v = (v_0, \ldots, v_{k-1})$ where $0 \leq v_i \leq n + \lceil (k-m)/m \rceil$ because after time n we need at most $\lceil (k-m)/m \rceil$ time units to satisfy the pending requests. The sub-problem corresponding to v gives the minimum total flow time for satisfying all the requests between v_{min} and n, i.e., the r_{ij} requests for P_j at time i for $v_{min} \leq i \leq n$ and $0 \leq j \leq k-1$ with v_j being P_j's earliest broadcasting time. Since we assume that an optimal schedule has m page broadcasts at each time slot, in particular the first time slot v_{min}, it is sufficient to consider only those vectors v corresponding to the sub-problems with m earliest page broadcasts at time v_{min}, i.e., there are exactly m v_j's values equal to v_{min}. Same as that in Section 2.2, we consider that every vector v has one of the v_j equal to $*$. For a vector v with $v_\alpha = *$, the possible values between $v_{min}+1$ and $n + \lceil (k-m)/m \rceil$ that can be assigned to v_α are in $C_v = \{i \mid$ there are less than m v_j's values equal to $i\}$.

To compute $F(v)$ of a vector v with $v_\alpha = *$, we find the minimum total flow time among the sub-problems corresponding to v with v_α assigned value β for each $\beta \in C_v$. For each of these sub-problems, if $v_{x_1} = v_{x_2} = \ldots = v_{x_m} = v_{min}$ for some $0 \leq x_1, \ldots, x_m \leq k-1$, then the optimal schedule must have all pages $P_{x_1}, \ldots, P_{x_m}$ broadcast at each of times $v_{min}, v_{min}+1, \ldots, s-1$ where $s = \min\{\beta, \min_{v_j \neq v_{min} \ \& \ v_j \neq *}\{v_j\}\}$ is the earliest broadcasting time among the pages other than $P_{x_1}, \ldots, P_{x_m}$. Note that there is no pending request for $P_{x_1}, \ldots, P_{x_m}$ immediately after time $s-1$. Thus, each of these sub-problems depends on one "smaller" sub-problem which corresponds to a *relaxed* k-dimensional vector $\tilde{u}$ in which exactly m $\tilde{u}_j$'s values equal to $*$. We construct $\tilde{u}$ as follows.

$$\tilde{u}_i = \begin{cases} * & \text{for } i \text{ where } v_i = v_{min}, \\ \beta & \text{for } i \text{ where } v_i = *, \\ v_i & \text{otherwise.} \end{cases}$$

Let $\tilde{u}_{min} = \min_{\tilde{u}_j \neq *}\{\tilde{u}_j\}$. The sub-problem corresponding to $\tilde{u}$ is to find the minimum total flow time, say $F(\tilde{u})$, to satisfy all requests between $\tilde{u}_{min}$ and n, i.e., the r_{ij} requests for P_j at time i for $u_{min} \leq i \leq n$ and $0 \leq j \leq k-1$, with $\tilde{u}_j$ being P_j's earliest broadcasting time for $\tilde{u}_j \neq *$. We do not need to compute $F(\tilde{u})$ directly. In fact $F(\tilde{u}) = \min\{F(v) \mid \tilde{u}_{min} = v_{min}$ and $\tilde{u}_j = v_j$ for all $\tilde{u}_j \neq *\}$, which is one of $F(v')$ for which the two sub-problems, corresponding to $\tilde{u}$ and v', respectively, have all pages having the same corresponding earliest broadcasting times except those pages P_j with $\tilde{u}_j = *$. We define the recurrence of $F(v)$ as follows.

$$F(v) = \min_{\text{sub-problems } \tilde{u} \text{ derived from } v} \{F(\tilde{u}) + c(\tilde{u}, v)\}$$

where $c(\tilde{u}, v) = \sum_{i=0}^{k} \sum_{t=v_{min}}^{\tilde{u}_{min}-1} f_{ti}(\tilde{u}, v)$ is the total flow time of the r_{ti} requests for P_i at time t for $0 \leq i \leq k-1$ and $v_{min} \leq t \leq \tilde{u}_{min} - 1$ and $f_{ti}(\tilde{u}, v)$ is the flow time of the particular r_{ti} requests for P_i at time t, i.e.,

$$f_{ti}(\tilde{u}, v) = \begin{cases} r_{ti} & \text{for } i \text{ where } v_i = v_{min}, \\ r_{ti}(\tilde{u}_i - t + 1) & \text{otherwise.} \end{cases}$$

We give an analysis on a brute-force implementation in solving the above recurrence of $F(v)$, and then we show a faster implementation. Lemma 5 implies that there are $O(k(n + k/m)^{k-m})$ different sub-problems we need to consider.

Lemma 5. *There are at most $O(k(n + k/m)^{k-m})$ different k-dimensional vectors $v = (v_0, \ldots, v_{k-1})$ satisfying the following conditions: (i) For all $0 \leq i \leq k-1$, $0 \leq v_j \leq n + k - 1$ except that one v_j's value is equal to $*$; (ii) for each $0 \leq t \leq n$, there are at most m v_j's values equal to t; and (iii) exactly m out of k v_j's values equal v_{min}.*

Proof (Sketch). Consider those vectors with that $v_0 = *$. As there are m v_j's values equal to v_{min}, the number of different vectors is bounded by the number of ways in choosing $k - m$ values, not necessarily distinct, between 0 and $n + \lceil (k-m)/m \rceil$, which is $O((n+k/m)^{k-m})$. Therefore the total number of different vectors is at most $O(k(n + k/m)^{k-m})$.

If all $F(\tilde{u})$ are known and can be retrieved in constant time, then each computation of $F(v)$ takes $O(kn)$ time because computing $c(\tilde{u}, v)$ takes $O(k)$ time and there are $O(n)$ different $\tilde{u}$ to be considered. We can compute all $F(\tilde{u})$ as follows. Since $F(\tilde{u}) = \min\{F(v) \mid \tilde{u}_{min} = v_{min}$ and $\tilde{u}_j = v_j$ for all $\tilde{u}_j \neq *\}$, after each $F(v)$ is computed we update the corresponding values of $F(\tilde{u})$ where $\tilde{u}_{min} = v_{min}$ and $\tilde{u}_j = v_j$ for all $\tilde{u}_j \neq *$, if $F(v) < F(\tilde{u})$. It takes $O(\binom{k-1}{k-m})$ time to update each $F(v)$ because there are $\binom{k-1}{k-m}$ corresponding $\tilde{u}$, as shown in the Lemma 6. Therefore, it takes $O(kn + \binom{k-1}{k-m})$ time to compute each $F(v)$, hence $O((kn + \binom{k-1}{k-m}))k(n + k)^{k-m})$ time to compute all values of $F(v)$ in the brute-force implementation.

Lemma 6. *For a k-dimensional vector v with exactly one v_j's value equals to $*$, there are $\binom{k-1}{k-m}$ k-dimensional (relaxed) vectors $\tilde{u}$ with exactly m $\tilde{u}_j$'s values equal to $*$ where $\tilde{u}_{min} = v_{min}$ and $\tilde{u}_j = v_j$ for all $\tilde{u}_j \neq *$.*

Proof (Sketch). It is equivalent to choosing $k-m$ of the $k-1$ v_j's values with $v_j \neq *$.

Similar to the efficient implementation in Section 2.2, we can partition the sub-problems $F(v)$ into groups and sub-groups so that we can apply the algorithm of Galil and Park [6] to compute all $F(v)$ of v corresponding to the sub-problems in a sub-group in $O(kn)$ time. Thus the overall time complexity for computing all $F(v)$ of v corresponding to the sub-problems of all sub-groups in all groups is $O((k^3 + \binom{k-1}{k-m})(n+k/m)^{k-m})$. When k and m are constants, the time complexity becomes $O(n^{k-m})$.

Theorem 4. *The minimum total flow time of the k-page broadcast scheduling problem with m broadcast channels where requests arriving at integral time 0 to time n can be computed in $O((k^3 + \binom{k-1}{k-m})(n+k/m)^{k-m})$ time, or $O(n^{k-m})$ time if k and m are constants.*

3 Approximation Algorithms

We present an approximation algorithm for the problem in minimizing the average flow time for the case when the number of broadcast channels $m = 1$. Note that the average flow time of a schedule equals the total flow time of the schedule divided by the total number of requests. Assuming that there are k pages that can be requested by clients, we give a simple algorithm with an additive approximation ratio of $(k-1)/2$.

When there are only two pages, P_0 and P_1, for clients to request, i.e., $k = 2$. The algorithm considers two particular schedules, $S_0 = (P_0, P_1, P_0, P_1, \ldots)$ and $S_1 = (P_1, P_0, P_1, P_0, \ldots)$. We show in Lemma 7 that either S_0 or S_1 has the average flow time at most 1/2 more than that of the optimal schedule. Thus, the algorithm achieves an additive approximation ratio of 1/2 by choosing among S_0 and S_1 the schedule with a smaller total flow time.

Lemma 7. *Either S_1 or S_2 has the average flow time at most* $1/2$ *more than that of the optimal schedule.*

Proof. Let r_{i0} and r_{i1} be the number of requests for P_0 and P_1 at time $0 \leq i \leq n$, respectively. Let $R = \sum_{i=0}^{n}(r_{i0} + r_{i1})$ be the total number of requests. The average flow time of S_0 is $T(S_0) = 1 + W_0/R$ where $W_0 = \sum_{i=1}^{n} r_{i,(i+1) \bmod 2}$ and the average flow time of S_1 is $T(S_1) = 1 + W_1/R$ where $W_1 = \sum_{i=0}^{n} r_{i,i \bmod 2}$. Since each page broadcast requires one flow time unit, the minimum average flow time $T^* \geq 1$. As $R = W_0 + W_1$, and thus $\min\{W_0/R, W_1/R\} \leq 1/2$ and $\min\{T(S_0), T(S_1)\} \leq T^* + 1/2$.

For the problem of k pages $P_0, P_1, \ldots, P_{k-1}$, we consider the following k schedules which broadcast each page cyclically with different starting pages: $S_0 = (P_0, P_1, \ldots, P_{k-1}, P_0, P_1, \ldots, P_{k-1}, \ldots)$, $S_1 = (P_1, \ldots, P_{k-1}, P_0, P_1, \ldots, P_{k-1}, P_0, \ldots)$, $\ldots$ $S_{k-1} = (P_{k-1}, P_0, \ldots, P_{k-2}, P_{k-1}, P_0, \ldots, P_{k-2}, \ldots)$. Again the approximation algorithm is to find the minimum average flow time schedule among these k schedules.

Theorem 5. *The minimum average flow time among schedules $S_0, \dots, S_{k-1}$ is at most $(k-1)/2$ more than that of the optimal schedule*

Proof. Let $W_i = \sum_{i=0}^{n} r_{i,(i+j) \bmod k}$ for $0 \leq j \leq k-1$ and $R = \sum_{i=0}^{k-1} W_i$ be the total number of requests. The average flow time of S_j for $0 \leq j \leq k-1$ is $T(S_j) = 1 + \sum_{i=1}^{k-1}(i \cdot W_{(i+j) \bmod k})/R$, and the minimum among them is at most $\sum_{j=0}^{k-1} T(S_j)/k = 1 + \sum_{j=0}^{k-1} \sum_{i=0}^{k-1} W_{(i+j) \bmod k}/(kR) = 1 + k(k-1) \sum_{j=0}^{k-1} W_j/(2kR) \leq T^* + (k-1)/2$, where $T^* \geq 1$ is the average flow time of the optimal algorithm.

References

1. N. Bansal, S. K. M. Charikar, and J. Naor. Approximating the aversage response time in broadcast scheduling. In *SODA 2005*.
2. Y. Bartal and S. Muthukrishnan. Minimizing maximum response time in scheduling broadcasts. In *SODA 2000*, pages 558–559.
3. J. Edmonds and K. Pruhs. Multicast pull scheduling: When fairness is fine. *Algorithmica*, 36(3):315–330, 2003.
4. J. Edmonds and K. Pruhs. A maiden analysis of longest wait first. In *SODA 2004*, pages 818–827.
5. T. Erlebach and A. Hall. Np-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. In *SODA 2002*, pages 194–202.
6. Z. Galil and K. Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Inf. Process. Lett.*, 33(6):309–311, 1990.
7. R. Gandhi, S. Khuller, Y. A. Kim, and Y.-C. J. Wan. Algorithms for minimizing response time in broadcast scheduling. *Algorithmica*, 38(4):597–608, 2004.
8. R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *FOCS 2002*, pages 323–332.
9. B. Kalyanasundaram, K. R. Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. *Journal of Scheduling*, 4(6):339–354, 2001.

Appendix

Lemma 8. *Given a sequence of $n+1$ numbers, $a_0, \dots, a_n$, with $O(n)$ time preprocessing, we can compute $\sum_{k=i}^{j} a_k$ for any $0 \leq i \leq j \leq n$ in constant time.*

Proof. As all the prefix sums $b_i = \sum_{k=0}^{i} a_k$ can be computed in $O(n)$ time, each of the partial sums $\sum_{k=i}^{j} a_k = b_j - b_{i-1}$ can be computed in constant time.

Lemma 9. *Given a sequence of $n+1$ numbers $a_0, \dots, a_n$, with $O(n)$ time preprocessing, we can compute $\sum_{k=i}^{j} a_k(d-k+1)$ for any $0 \leq i \leq j \leq n$ and $j \leq d$ in constant time.*

Proof. As all the prefix sums $b_i = \sum_{k=0}^{i} a_k$ and weighted prefix sums $w_i = \sum_{k=0}^{i} a_k(n-i+1)$ can be computed in $O(n)$ time, each of the functions $\sum_{k=i}^{j} a_k(d-k+1) = w_j - w_{i-1} + (d-n)(b_j - b_{i-1})$, can be computed in constant time.