A Dynamic Programming Approach of Finding an Optimal Broadcast Schedule in Minimizing Total Flow Time[*]

Wun-Tat Chan[†] Francis Y. L. Chin [‡] Yong Zhang [§] Hong Zhu[¶]
Hong Shen[‖] Prudence W.H. Wong[**]

Abstract

We study the problem of (off-line) broadcast scheduling in minimizing total flow time and propose a dynamic programming approach to compute an optimal broadcast schedule. Suppose the broadcast server has k pages and the last page request arrives at time n. The optimal schedule can be computed in $O(k^3(n+k)^{k-1})$ time for the case that the server has a single broadcast channel. For m channels case, i.e., the server can broadcast m different pages at a time where $m < k$, the optimal schedule can be computed in $O(n^{k-m})$ time when k and m are constants. Note that this broadcast scheduling problem is NP-hard when k is a variable and will take $O(n^{k-m+1})$ time when k is fixed and $m \geq 1$ with the straightforward implementation of the dynamic programming approach.

Keywords: Broadcast scheduling, flow time minimization, dynamic programming

[*]The preliminary version of this paper appeared in Proceedings of the 11th Annual International Computing and Combinatorics Conference as "Off-line Algorithms for Minimizing the Total Flow Time in Broadcast Scheduling".

[†]Department of Computer Science, University of Hong Kong, Pokfulam Rd., Hong Kong, wtchan@cs.hku.hk. This research was supported in part by Hong Kong RGC Grants HKU5172/03E.

[‡]Department of Computer Science, University of Hong Kong, Pokfulam Rd., Hong Kong, chin@cs.hku.hk. This research was supported in part by Hong Kong RGC Grants HKU7142/03E.

[§]Department of Computer Science, University of Hong Kong, Pokfulam Rd., Hong Kong, yzhang@cs.hku.hk.

[¶]Department of Computer Science and Engineering, Fudan University, China, hzhu@fudan.edu.cn.

[‖]Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Japan, shen@jaist.ac.jp.

[**]Department of Computer Science, University of Liverpool, UK, pwong@csc.liv.ac.uk. This research was supported in part by Nuffield Foundation Grant NAL/01004/G.

1 Introduction

In an on-demand broadcasting system, the server receives the client requests for pages at arbitrary times and serves the requests by broadcasting (sending) pages via the broadcast channels. After the server broadcasts a page, all pending requests for that page are satisfied. The goal of the broadcast scheduler is to arrange the order of page broadcasts so as to minimize the total (or average) flow time of the requests.

In this paper we assume that time is discrete and represented by non-negative integers. Every page can be broadcast to clients in one time unit. The server has m broadcast channels, i.e., at most m different pages can be broadcast at a time. The broadcast scheduling problem is formulated as follows. Assume that the server contains k pages, namely $P_0, P_1, \ldots, P_{k-1}$. The requests for these pages arrive at some integer times. Let $r_{t,i}$ denote the number of requests for P_i at time t. For a schedule, let $b_{t,i}$ be the earliest time at or after time t when P_i is broadcast. The *flow time* of a request for P_i arriving at time t is $b_{t,i} - t + 1$. Suppose the last request arrives at time n. The total flow time of the schedule, which is to be minimized, is equal to $\sum_{t=0}^{n} \sum_{i=0}^{k-1} r_{t,i}(b_{t,i} - t + 1)$. Note that an optimal schedule that minimizes the total flow time is also an optimal schedule that minimizes the 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. Erlebach and Hall [6] showed that this problem is NP-hard. Bansal et al. [1] gave an algorithm that achieved an $O(\sqrt{k})$ approximation, and very recently the algorithm was improved with an approximation factor $O(\log^2(k+n))$ [2]. Besides, most of the previous works considered the resource augmentation setting. An m-speed algorithm refers to an algorithm that utilizes m broadcast channels and an m-speed c-approximation algorithm is an m-speed algorithm that produces schedules with total flow time at most c times that of the schedule by the optimal 1-speed algorithm. Kalyanasundaram et al. [10] gave an $\frac{1}{\epsilon}$-speed $\frac{1}{1-2\epsilon}$-approximation algorithm for any fixed $\epsilon \in (0, \frac{1}{3}]$. Gandhi et al. [8] gave an $\frac{1}{\epsilon}$-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 [6] gave a 6-speed algorithm, which was improved to 4-speed [8] and then to 3-speed by Gandhi et al. [9]. The on-line version of the problem was studied by Edmonds and Pruhs [4, 5]. Bartal and Muthukrishnan [3] considered the problem that minimizes the maximum flow time.

This paper is the first to give algorithms for finding optimal broadcast schedules that minimize total flow time. Note that the straightforward implementation of the dynamic programming approach will take $O(n^k)$ time. Based on a dynamic programming technique and the concave property of the optimization function, our algorithms construct an optimal schedule for the case when $m = 1$ in $O(k^3(n+k)^{k-1})$ time where k is the number of pages and the last request arrives at time n. When k is a constant, the time complexity is $O(n^{k-1})$. We generalize this result in the m channels case where a server has $1 \le m < k$ broadcast channels. We show that an optimal schedule can be found in $O(k\binom{k-1}{m}(\binom{k-1}{m-1}+k)(n+k/m)^{k-m})$ time, or $O(n^{k-m})$ time when k and m are constants.

The rest of the paper is organized as follows. In order to illustrate the idea, a simple case when $m = 1$ and $k = 2$ is considered. A straightforward dynamic programming implementation would take $O(n^2)$ time. In Section 2, a linear-time optimal algorithm based on the concavity of the optimization function for $m = 1$ and $k = 2$ is given. The application of the concave property for general k is not straightforward, we have shown, in Sections 3 and 4 respectively, how the algorithms for the cases of $m = 1$ and general m can be speeded up.

2 Broadcast Scheduling for Two Pages

Assume we have $m = 1$ broadcast channel, and $k = 2$ pages (P_0 and P_1), and the last request arrives at time n. Our target is to efficiently compute the minimum total flow time for satisfying all requests.

Defintition 1. *For $i \in \{0, 1\}$ and $0 \le t \le n+1$, let $F_i(t)$ denote the minimum total flow time in satisfying all the requests arriving at or after time t where P_i must be broadcast at time t.*

Note that $F_i(n + 1) = 0$ because there is no request after time n. As either P_0 or P_1 is broadcast at time 0, we can see that the minimum total flow time in satisfying all requests, denoted by F, is equal to $\min\{F_0(0), F_1(0)\}$. In the following we show how $F_i(t)$ can be computed recursively. The base case is when $t = n + 1$,

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

In general, 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 $c_1(s,t) = \sum_{i=t}^{s-1} (r_{i,0} + r_{i,1}(s - i + 1))$ is the total flow time in satisfying the requests arriving between time t and time $s - 1$ inclusively. Similarly,

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

where $c_0(s,t) = \sum_{i=t}^{s-1} (r_{i,0}(s - i + 1) + r_{i,1})$.

By Lemmas 2 and 3, functions $c_0(s,t)$ and $c_1(s,t)$ can be computed in constant time for any given s and t after an $O(n)$-time preprocessing.

Lemma 2. *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 $0 \le i \le j \le n$ in constant time.*

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

Lemma 3. *Given a sequence of $n+1$ numbers $a_0, \dots, a_n$, with $O(n)$ time preprocessing, we can compute $\sum_{k=i}^{j} a_k(d-k)$ for any $0 \le i \le j \le n$ and any d in constant time.*

Proof. Compute all the prefix sums $b_i = \sum_{k=0}^{i} a_k$ and weighted prefix sums $w_i = \sum_{k=0}^{i}(k \cdot a_k)$ in $O(n)$ time. After that, each of the functions

$$\sum_{k=i}^{j} a_k(d-k) = d\sum_{k=i}^{j} a_k - \sum_{k=i}^{j}(k \cdot a_k) = d(b_j - b_{i-1}) - (w_j - w_{i-1}),$$

can be computed in constant time. □

Implementing the recursive formulas with a brute-force method, F can be found by computing all $F_i(t)$ for all $i \in \{0,1\}$ and $0 \le t \le n$, which takes $O(n^2)$ time. By the technique of Galil and Park [7], we show that F, as well as the optimal schedule, can be found in linear time. We say that a function $w()$ is *concave* if it satisfies the *quadrangle inequality*, i.e., $w(a,c) + w(b,d) \le w(a,d) + w(b,c)$ for $a \le b \le c \le d$. Galil and Park proved the following theorem.

Theorem 4 (Galil and Park [7])**.** *Given a concave function $w(i,j)$ for integer $0 \le i \le j \le n$ and given $E(0)$, 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 4, and thus they 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 $0 \le j \le n+1$. The base case is $E(0) = F(n+1) = 0$. 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 5 shows that function $w(i,j)$ satisfies the quadrangle inequality. Thus, by Theorem 4, we can find the minimum total flow time and the optimal schedule in linear time, as given in Theorem 6.

Lemma 5. *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., $w(a,c) + w(b,d) \le w(a,d) + w(b,c)$ for integer $a \le b \le c \le d$.*

Proof. We consider the case of $c_1(n-i+1, n-j+1)$ and the case of $c_0(n-i+1, n-j+1)$ can be proved similarly. For $w(i,j) = c_1(n-i+1, n-j+1) = \sum_{x=n-j+1}^{n-i}(r_{x,0}+r_{x,1}(n-i-x+2))$, we can see that $\sum_{x=n-c+1}^{n-a} r_{x,0} + \sum_{x=n-d+1}^{n-b} r_{x,0} = \sum_{x=n-d+1}^{n-a} r_{x,0} + \sum_{x=n-c+1}^{n-b} r_{x,0}$ and

$$\begin{aligned}
& \sum_{x=n-c+1}^{n-a} r_{x,1}(n-a-x+2) + \sum_{x=n-d+1}^{n-b} r_{x,1}(n-b-x+2) \\
= \; & \sum_{x=n-c+1}^{n-a} r_{x,1}(n-a-x+2) + \sum_{x=n-d+1}^{n-c} r_{x,1}(n-b-x+2) + \sum_{x=n-c+1}^{n-b} r_{x,1}(n-b-x+2) \\
\leq \; & \sum_{x=n-d+1}^{n-a} r_{x,1}(n-a-x+2) + \sum_{x=n-c+1}^{n-b} r_{x,1}(n-b-x+2). \qquad \{\because n-a \geq n-b\}
\end{aligned}$$

The last inequality is due to $n-a \geq n-b$. Therefore we have $w(a,c) + w(b,d) \leq w(b,c) + w(a,d)$. □

Theorem 6. *An optimal schedule in minimizing the total flow time for the broadcast scheduling problem with one channel and two pages and requests arriving at integer time* 0 *to time* n *can be computed in* $O(n)$ *time.*

3 Broadcast Scheduling for k Pages

In this section we consider the problem with a single broadcast channel and k pages, in particular for $k \geq 2$. We formulate the problem as a dynamic programming problem which is a generalization of that in Section 2.

> A sub-problem in the dynamic programming can be specified by a k-dimensional vector $v = (v_0, \ldots, v_{k-1})$. A value v_i represents the earliest broadcast time of P_i, and hence $0 \leq v_i \leq n+k-1$. Moreover, at any time only one page is broadcast, i.e., $v_i \neq v_j$ if $i \neq j$.

The sub-problem corresponding to v is to find the minimum total flow time in satisfying all the requests arriving between $\min_{0\leq i\leq k-1}\{v_i\}$ and n inclusively, with v_i being the earliest broadcast time P_i. For example, when $k = 2$, $F_0(t)$ defined in Section 2 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 $\Omega((n+k)^k)$ possible such k-dimensional vectors as well as sub-problems, the time complexity in solving the recurrence will be at least $\Omega((n+k)^k)$. In the following, we modify slightly the definition of the vectors corresponding to the sub-problems 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 "$*$". If $v_\alpha = *$ for some $0 \leq \alpha < k$, it means that in the sub-problem corresponding to v, the earliest broadcast time of P_α is not fixed, yet it cannot be earlier than that of all other pages.

Defintition 7. *For a vector $v = (v_0, \ldots, v_{k-1})$, denote by v_{min} the minimum values among v_i besides that equals to $*$. Precisely, $v_{min} = \min_{0 \le j \le k-1 \,\&\, v_j \ne *}\{v_j\}$.*

The sub-problem corresponding to v, say with $v_\alpha = *$, is to find the minimum total flow time in satisfying all the requests arriving between time v_{min} and n inclusively, with v_i being the earliest broadcast time of P_i for $i \ne \alpha$. The earliest broadcast time of P_α can be any possible value between $v_{min}+1$ and $n+k-1$ which is not equal to any other v_i, i.e., some integer in $C_v = \{t \mid t \ne v_j$ for all $v_j \ne *$ and $v_{min}+1 \le t \le n+k-1\}$.

Defintition 8. *Let $F(v)$ denote the minimum total flow time for the sub-problem corresponding to a vector v.*

The function $F(v)$ can be defined recursively as follows. In the base case, we let

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

because there is no request arriving after time n. In general, we consider $0 \le v_{min} \le n$ and assume that $v_\alpha = *$. Although v_α is unspecified, the allowable earliest broadcast time of P_α can only be some value $\beta \in C_v$. Therefore, $F(v)$ equals the minimum total flow time among the sub-problems corresponding to v with v_α assigned a value β, for each $\beta \in C_v$. After a value of β is chosen, we observed a property similar to that of the recurrence in Section 2. If $v_x = v_{min}$, then the schedule must have P_x broadcast at each time $v_{min}, v_{min}+1, \ldots, s-1$ where $s = \min\{\beta, \min_{v_j \ne v_{min} \,\&\, v_j \ne *}\{v_j\}\}$ is the earliest broadcast time of the pages other than P_x. Note that there is no pending request for P_x immediately after time $s-1$. Thus, we can identify a "smaller" sub-problem corresponding to a vector $v^\beta = (v_0^\beta, \ldots, v_{k-1}^\beta)$ based on v and β as follows.

$$v_i^\beta = \begin{cases} * & \text{for } i \text{ where } v_i = v_{min}, \\ \beta & \text{for } i = \alpha, \text{ i.e., } v_i = *, \\ v_i & \text{otherwise.} \end{cases} \tag{1}$$

Let $f_{t,i}(v^\beta, v)$ be the total flow time of the $r_{t,i}$ requests for P_i arriving at time t for $v_{min} \le t \le v_{min}^\beta - 1$, i.e.,

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

The total flow time of the $r_{t,i}$ requests for P_i arriving at time t for $v_{min} \le t \le v_{min}^\beta - 1$ and $0 \le i \le k-1$, denoted by $c(v^\beta, v)$, is $\sum_{i=0}^{k-1} \sum_{t=v_{min}}^{v_{min}^\beta - 1} f_{t,i}(v^\beta, v)$. To compute $F(v)$, we can consider the $|C_v|$ different "smaller" sub-problems resulting from choosing the $|C_v|$ different values for β, i.e.,

$$F(v) = \min_{\beta \in C_v}\{F(v^\beta) + c(v^\beta, v)\}. \tag{2}$$

3.1 Straightforward Implementation

First, we give an analysis on a brute-force implementation in solving the above recurrence of $F(v)$. Then we present a faster implementation by generalizing the approach used in Section 2. Similar to the case of $k = 2$ in Section 2, with $O(kn)$-time preprocessing, $\sum_{t=v_{min}}^{v_{min}^{\beta}-1} f_{t,i}(v^{\beta}, v)$ can be computed in constant time for any given i, v_{min} and $v_{min}^{\beta} - 1$ (see Lemmas 2 and 3) and thus a particular $c(v^{\beta}, v)$ can be computed in $O(k)$ time. Since the number of sub-problems corresponding to v^{β}, derived from a given v, is $O(n)$, a particular $F(v)$ can be computed in $O(kn)$ time. Lemma 9 implies that there are $k(n + k)!/(n + 1)!$ different sub-problems corresponding to a k-dimensional vector. Therefore, the brute-force method in finding the minimum total flow time by computing all $F(v)$ takes $O(k^2 n(n + k)!/(n + 1)!)$ time, or concisely, $O(k^2(n + k)^k)$ time.

Lemma 9. *There are $k(n+k)!/(n+1)!$ different k-dimensional vectors $v = (v_0, \ldots, v_{k-1})$ with $v_i \in \{*\} \cup \{0, \ldots, n - k + 1\}$ for $0 \le i \le k - 1$ and exactly one of v_i's value must equal $*$ and $v_i \neq v_j$ for $i \neq j$.*

Proof. As each v_i should have a distinct value and one of them must be $*$, there are $\binom{n+k}{k-1}$ ways of choosing the $k - 1$ distinct values from $\{0, 1, \ldots, n + k - 1\}$. Since the $k - 1$ distinct values have $k!$ ways of assigning to the k positions of v_i, there are altogether $k(n + k)!/(n + 1)!$ different possible vectors of v. □

3.2 Efficient Implementation

Note that up to this stage, there is no gain in the time complexity and $\Omega((n + k)^k)$ time is still needed. In order to improve the time complexity, the concave property of the function $c(v^{\beta}, v)$ should be exploited as in Section 2. In the faster implementation, we group the sub-problems, equivalently the corresponding vectors, into k^2 groups denoted as $G_{x,y}$ for $0 \le x, y \le k - 1$. Two vectors v and u belong to the same group $G_{x,y}$ if $v_{min} = v_x$ and $u_{min} = u_x$, and $v_y = u_y = *$, i.e., both sub-problems each corresponding to v and u having P_x broadcast earlier than all the other pages and the earliest broadcast time of P_y unspecified. We further divide the vectors in each group into sub-groups. Two vectors v and v' of the same group $G_{x,y}$ belong to the same sub-group if except P_x and P_y every page have the same earliest broadcast time in both sub-problems corresponding to v and v', i.e., $v_j = v'_j$ for all j with $j \neq x$ and $j \neq y$. It is clear that there are $O(n)$ vectors in each sub-group and we can prove that there are $(n+k-1)!/(n+1)!$ sub-groups in each group in the following lemma.

Lemma 10. *There are $(n + k - 1)!/(n + 1)!$ sub-groups in each group.*

Proof. By symmetry, there are the same number of sub-groups in each group. Without loss of generality, we can consider a particular group $G_{0,1}$. For $v \in G_{0,1}$, since v_0 is the minimum among all v_j except v_1, no v_j for $2 \leq j \leq k-1$ can be of value 0. The number of sub-groups in $G_{0,1}$ is equal to the number of ways of choosing $k-2$ distinct values from $\{1, 2, \ldots, n+k-1\}$ for $v_2, v_3, \ldots, v_{k-1}$, i.e., $\binom{n+k-1}{k-2}$. As these $k-2$ distinct values have $(k-2)!$ ways of assigning to $v_2, v_3, \ldots, v_{k-1}$, there are $(n+k-1)!/(n+1)!$ sub-groups in $G_{0,1}$, and each other group. □

Consider the set of $F(v)$ for all vectors v of the same sub-group. We can transform the recurrence for these $F(v)$ to the form as in Theorem 4 of Section 2. Without loss of generality, we consider a sub-group H from $G_{0,1}$, and other sub-groups can be handled similarly. For all vectors v in H, we have $v_0 = v_{min}$ and $v_1 = *$ and all other v_j fixed. For ease of explanation, we assume that $v_j \in \{n+2, n+3, \ldots, n+k-1\}$ for all $2 \leq j \leq k-1$. (The assumption is not necessary for the correctness of our algorithm.) For $0 \leq t \leq n+1$, let $E(t) = F(v)$ where $v_0 = v_{min} = n-t+1$. The base case is $E(0) = F(v) = 0$ when $t = 0$ and $v_0 = n+1$. For $0 \leq s \leq n$, let $u = v^{n-s+1}$ and $D(s) = F(u)$ where $u_0 = *, u_1 = n-s+1$, and $u_j = v_j$ for $2 \leq j \leq k-1$. The construction of u (i.e., v^{n-s+1}) from v follows that in Equation (1). Since the values v_j for $2 \leq j \leq k-1$ are fixed for all v in H, there are only $n+1$ values of $F(u)$ to consider. For $0 \leq s < t \leq n+1$, let $w(s,t) = c(u,v)$, which is given as follows.

$$w(s,t) = \sum_{i=n-t+1}^{n-s} r_{i,0} + \sum_{i=n-t+1}^{n-s} r_{i,1}(n-s-i+2) + \sum_{j=2}^{k-1} \sum_{i=n-t+1}^{n-s} r_{i,j}(v_j - i + 1)$$

Lemma 11 shows that the function $w(s,t)$ satisfies the quadrangle inequality. By Theorem 4, all $F(v)$ for v in H can be computed in $O(kn)$ time. Together with Lemma 10 and the fact that there are k^2 groups, all $F(v)$ of all sub-groups can be computed in $O(kn \cdot k^2 \cdot (n+k-1)!/(n+1)!)$, or concisely, $O(k^3(n+k)^{k-1})$ time. Thus, we have Theorem 12.

Lemma 11. *The function $w(s,t)$ for $0 \leq s < t \leq n+1$ satisfies the quadrangle inequality, i.e., $w(a,c) + w(b,d) \leq w(a,d) + w(b,c)$ for $a \leq b \leq c \leq d$.*

Proof. We have

$$w(s,t) = \sum_{i=n-t+1}^{n-s} \left(\left(r_{i,0} + \sum_{j=2}^{k-1} r_{i,j}(v_j - i + 1) \right) + r_{i,1}(n-s-i+2) \right).$$

Consider $r_{i,0} + \sum_{j=2}^{k-1} r_{i,j}(v_j - i + 1)$ as a function of i, say $f(i)$. It is easy to see that $\sum_{i=n-c+1}^{n-a} f(i) + \sum_{i=n-d+1}^{n-b} f(i) = \sum_{i=n-d+1}^{n-a} f(i) + \sum_{i=n-c+1}^{n-b} f(i)$. As in the proof of Lemma 5, we have shown that $\sum_{i=n-c+1}^{n-a} r_{i,1}(n-a-i+2) + \sum_{i=n-d+1}^{n-b} r_{i,1}(n-b-i+2) \leq \sum_{i=n-d+1}^{n-a} r_{i,1}(n-a-i+2) + \sum_{i=n-c+1}^{n-b} r_{i,1}(n-b-i+2)$. Hence, $w()$ satisfies the quadrangle inequality. □

Theorem 12. *An optimal schedule in minimizing the total flow time for the broadcast scheduling problem with one channel and k pages and requests arriving at integer time 0 to time n can be computed in $O(k^3(n+k)^{k-1})$ time.*

4 Broadcast Scheduling with Multiple Channels

Assume that there are m broadcast channels available to the server. At each time slot, the server can broadcast at most m different pages among the k pages, where $m < k$. Without loss of generality, we can assume that there is an optimal schedule that broadcasts exactly m different pages at each time slot.

We apply the framework in Section 3 to solve the problem using the dynamic programming approach. Each sub-problem in the dynamic programming can be specified by a k-dimensional vector $v = (v_0, \ldots, v_{k-1})$ where $0 \leq v_i \leq n + \lceil k/m \rceil - 1$ represents the earliest broadcast time of P_i. It is clear that after time n we need at most $\lceil k/m \rceil - 1$ time units to satisfy all pending requests. Let $t = \min_{1 \leq i \leq k}\{v_i\}$. The sub-problem corresponding to v is to find the minimum total flow time in satisfying all the requests arriving between t and n inclusively, with v_i being the earliest broadcast time of P_i. Since we consider the schedules that broadcast m pages at each time, in particular time t, it is sufficient to consider only those vectors v with m v_j's values equal to t.

Same as that in Section 3, we consider every vector v has one of the v_j equal to $*$. For a vector v with $v_\alpha = *$ for some $0 \leq \alpha < k$, it means that in the corresponding sub-problem the earliest broadcast time of P_α is unspecified but it can only be some integer in $C_v = \{i \mid \text{there are less than } m \ v_j\text{'s values equal to } i \text{ for } 0 \leq i \leq n + \lceil k/m \rceil - 1\}$.

We adopt the notations defined in Section 3. Let $F(v)$ denote the minimum total flow time for the sub-problem corresponding to v and let $v_{min} = \min_{1 \leq i \leq k \ \& \ v_i \neq *}\{v_i\}$. We show that $F(v)$ can be defined recursively. For the base case, $F(v) = 0$ if $v_{min} \geq n + 1$. In general, we assume $0 \leq v_{min} \leq n$ and $v_\alpha = *$. Consider a schedule S for the sub-problem corresponding to v. Let β be the earliest broadcast time of P_α. It is clear that $\beta \in C_v$. Suppose $v_{x_1} = v_{x_2} = \ldots = v_{x_m} = v_{min}$ for some $0 \leq x_1, \ldots, x_m \leq k-1$. S 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 broadcast time of 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 $s - 1$.

We can construct a "smaller" sub-problem based on v and β. This sub-problem is characterized by another vector, denoted by $\tilde{v}^\beta$, which is *relaxed* in the sense that exactly

m $\tilde{v}_j^\beta$'s values equal to $*$ and the vector is defined as follows.

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

Similar to Definition 7, let $\tilde{v}_{min}^\beta = \min_{0 \le j \le k-1 \text{ \& } \tilde{v}_j^\beta \ne *} \{\tilde{v}_j^\beta\}$. The sub-problem corresponding to $\tilde{v}^\beta$ is to find the minimum total flow time, denoted by $F(\tilde{v}^\beta)$, in satisfying all requests arriving between $\tilde{v}_{min}^\beta$ and n inclusively, with $\tilde{v}_j^\beta$ being the earliest broadcasting time of P_j for $\tilde{v}_j^\beta \ne *$. We do not need to compute $F(\tilde{v}^\beta)$ directly. In fact $F(\tilde{v}^\beta) = \min\{F(u) \mid u$ is a non-relaxed vector and $u_{min} = \tilde{v}_{min}^\beta$ and $u_j = \tilde{v}_j^\beta$ for all $\tilde{v}_j^\beta \ne *\}$, which is the minimum $F(u)$ among those corresponding to the sub-problems with the earliest broadcast time u_j of P_j at time $\tilde{v}_j^\beta$, for all j except those with $\tilde{v}_j^\beta = *$.

Let $f_{t,i}(\tilde{v}^\beta, v)$ be the flow time of the $r_{t,i}$ requests for P_i arriving at time t for $v_{min} \le t \le \tilde{v}_{min}^\beta - 1$, i.e.,

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

The total flow time of the $r_{t,i}$ requests for P_i arriving at time t for $v_{min} \le t \le \tilde{v}_{min}^\beta - 1$ and $0 \le i \le k-1$, denoted by $c(\tilde{v}^\beta, v)$, is $\sum_{i=0}^{k} \sum_{t=v_{min}}^{\tilde{v}_{min}^\beta - 1} f_{t,i}(\tilde{v}^\beta, v)$. The recurrence of $F(v)$ can be defined as follows,

$$F(v) = \min_{\beta \in C_v} \{F(\tilde{v}^\beta) + c(\tilde{v}^\beta, v)\}.$$

4.1 Straightforward Implementation

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 13 implies that there are $O(k\binom{k-1}{m}(n + k/m)^{k-m})$ different sub-problems we need to consider.

Lemma 13. *There are at most $O(k\binom{k-1}{m}(n + k/m)^{k-m})$ different k-dimensional vectors $v = (v_0, \ldots, v_{k-1})$ satisfying the following conditions: (i) For all $0 \le i \le k-1$, $v_j \in \{*\} \cup \{0, \ldots, n + \lceil k/m \rceil - 1\}$ and exactly one of v_j's value must equal $*$; (ii) for each $0 \le t \le n + \lceil k/m \rceil - 1$, there are at most m v_j's values equal to t; and (iii) exactly m v_j's values equal v_{min}.*

Proof. For the v_j for $1 \le j \le k$, there are $k\binom{k-1}{m}$ combinations such that one of them is chosen for $*$ and m of them are chosen for v_{min}. For the actual values of v_j, we have $0 \le v_{min} \le n + \lceil k/m \rceil - 1$ and $1 \le v_j \le n + \lceil k/m \rceil - 1$ for $v_j \ne v_{min}$ and $v_j \ne *$. Therefore the total number of different vectors is at most $O(k\binom{k-1}{m} \cdot (n + \lceil k/m \rceil) \cdot (n + \lceil k/m \rceil - 1)^{k-m-1})$, i.e., $O(k\binom{k-1}{m}(n + k/m)^{k-m})$. □

If $F(\tilde{v})$ for all relaxed vector $\tilde{v}$ are known and can be retrieved in constant time, then the computation of $F(v)$ for each non-relaxed vector v takes $O(k(n+k/m))$ time because computing $c(\tilde{v}^{\beta}, v)$ takes $O(k)$ time and there are $O(n+k/m)$ different $\tilde{v}^{\beta}$ to be considered. We can compute all $F(\tilde{v})$ as follows. Since $F(\tilde{v}) = \min\{F(v) \mid v$ is a non-relaxed vector and $v_{min} = \tilde{v}_{min}$ and $v_j = \tilde{v}_j$ for all $\tilde{v}_j \neq *\}$, after each $F(v)$ is computed we update the corresponding values of $F(\tilde{v})$ where $\tilde{v}_{min} = v_{min}$ and $\tilde{v}_j = v_j$ for all $\tilde{v}_j \neq *$, if $F(v) < F(\tilde{v})$. It takes $O(\binom{k-1}{m-1})$ time to update for each $F(v)$ because there are $\binom{k-1}{m-1}$ corresponding $\tilde{v}$, as shown in Lemma 14. Therefore, it takes $O(k(n+k/m)+\binom{k-1}{m-1})$ time to handle each $F(v)$, hence $O((k(n+k/m)+\binom{k-1}{m-1})k\binom{k-1}{m}(n+k/m)^{k-m})$ time to compute all values of $F(v)$ in the brute-force implementation. If k and m are constant, it still takes $O(n^{k-m+1})$ time.

Lemma 14. *For a k-dimensional non-relaxed vector v with exactly one v_j's value equals to $*$, there are $\binom{k-1}{m-1}$ k-dimensional relaxed vectors $\tilde{v}$ with exactly m $\tilde{v}_j$'s values equal to $*$ where $\tilde{v}_{min} = v_{min}$ and $\tilde{v}_j = v_j$ for all $\tilde{v}_j \neq *$.*

Proof. Since there is already one v_j's value equal to $*$, we need to select $m-1$ more of v_j to be $*$. Thus the number of $\tilde{v}$ is equivalent to the number of ways in choosing $m-1$ out of the $(k-1)$ v_j's values with $v_j \neq *$. □

4.2 Efficient Implementation

Similar to the efficient implementation in Section 3, we can partition the vectors into groups and sub-groups. Two vectors v and u belong to the same group $G(x_1, x_2, \ldots, x_m, y)$, if $v_{x_1} = v_{x_2} = \ldots = v_{x_m} = v_{min}$ and $u_{x_1} = u_{x_2} = \ldots = u_{x_m} = u_{min}$ and $v_y = u_y = *$. Furthermore, two vectors v and v' of the same group $G(x_1, x_2, \ldots, x_m, y)$ belong to the same sub-group if $v_j = v'_j$ for all j with $j \neq x_1, \ldots, j \neq x_m$, and $j \neq y$. Then by Theorem 4 we can compute all $F(v)$ of v in a sub-group in $O(k(n+k/m))$ time. The concave property of $c(\tilde{v}^{\beta}, v)$ can be proved as in Lemma 11. Theorem 16 shows that the overall time complexity of computing all $F(v)$ is $O(k\binom{k-1}{m}(\binom{k-1}{m-1}+k)(n+k/m)^{k-m})$. When k and m are constants, the time complexity becomes $O(n^{k-m})$.

Lemma 15. *There are $k\binom{k-1}{m}$ groups and there are at most $O((n+k/m)^{k-m-1})$ sub-groups in each group.*

Proof. The number of groups is equal to the number of ways in choosing two disjoint subsets $\{y\}$ and $\{x_1, x_2, \ldots, x_m\}$ from $\{1, \ldots, k\}$ with one and m values, respectively, which is $(m+1)\binom{k}{m+1} = k\binom{k-1}{m}$. The number of sub-groups in each group is $O((n+k/m)^{k-m-1})$ because for a vector v in $G(x_1, \ldots, x_2, y)$ each v_j can be assigned a value in $\{1, \ldots, n+\lceil k/m \rceil - 1\}$ for $j \neq x_1, \ldots, j \neq x_m$, and $j \neq y$, which consists of at most $O((n+\lceil k/m \rceil - 1)^{k-m-1})$, i.e., $O((n+k/m)^{k-m-1})$ different combinations. □

Theorem 16. *An optimal schedule in minimizing the total flow time for the broadcast scheduling problem with m channels and k pages and requests arriving at integer time 0 to time n can be computed in $O(k\binom{k-1}{m}(\binom{k-1}{m-1}+k)(n+k/m)^{k-m})$ time, or $O(n^{k-m})$ time if k and m are constants.*

Proof. By Lemmas 13 and 14, the overall time to update $F(\tilde{v})$ after computing each $F(v)$ is $O(k\binom{k-1}{m-1}\binom{k-1}{m}(n+k/m)^{k-m})$. By Lemma 15, the overall time to compute $F(v)$ using the technique of Galil and Park is $O(k(n+k/m)\cdot(n+k/m)^{k-m-1}\cdot k\binom{k-1}{m})$, i.e., $O(k^2\binom{k-1}{m}(n+k/m)^{k-m})$. Altogether the time complexity is $O(k\binom{k-1}{m}(\binom{k-1}{m-1}+k)(n+k/m)^{k-m})$, or $O(n^{k-m})$ if k and m are constants. $\square$

References

[1] N. Bansal, M. Charikar, S. Khanna, and J. Naor. Approximating the average response time in broadcast scheduling. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 215–221, 2005.

[2] N. Bansal, D. Coppersmith, and M. Sviridenko. Improved approximation algorithms for broadcast scheduling. In *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

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

[4] J. Edmonds and K. Pruhs. Multicast pull scheduling: When fairness is fine. *Algorithmica*, 36(3):315–330, 2003.

[5] J. Edmonds and K. Pruhs. A maiden analysis of longest wait first. *ACM Trans. Algorithms*, 1(1):14–32, 2005.

[6] T. Erlebach and A. Hall. NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *J. Scheduling*, 7(3):223–241, 2004.

[7] Z. Galil and K. Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Inf. Process. Lett.*, 33(6):309–311, 1990.

[8] 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.

[9] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *Proc. 43th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 323–332, 2002.

[10] B. Kalyanasundaram, K. R. Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. *J. Scheduling*, 4(6):339–354, 2001.