Windows Scheduling Problems for Broadcast Systems

Amotz Bar-Noy*　　Richard E. Ladner†

October 18, 2001

Abstract

The *windows scheduling problem* is defined by the positive integers h and $w_1, w_2, \ldots, w_n$. The *window* w_i is associated with *page* i and h is the number of *slotted channels* available for broadcasting the pages. A schedule that solves the problem assigns pages to slots such that the gap between any two consecutive appearances of page i is at most w_i slots. We investigate two optimization problems. (i) The *optimal windows scheduling problem*: given $w_1, \ldots, w_n$ find a schedule in which h is minimized. (ii) The *optimal harmonic windows scheduling problem*: given h find a schedule for the windows $w_i = i$ in which n is maximized. The former is a formulation of the problem of minimizing the bandwidth in push systems that support guaranteed delay and the latter is a formulation of the problem of minimizing the startup delay in media-on-demand systems. For the optimal windows scheduling problem we present an algorithm that constructs asymptotically close to optimal schedules and for the optimal harmonic windows scheduling problem, we show how to achieve the largest known n's for all values of h.

*AT&T research, Shannon Lab, 180 Park Ave., Florham Park, NJ 07932. E-mail: *amotz@research.att.com*. This work was done while the author was a member of the Electrical Engineering Systems Department, Tel Aviv University, Tel Aviv 69978, Israel.

†Department of Computer Science and Engineering, Box 352350, University of Washington, Seattle, WA 98195. This work was partially supported by NSF grant No. CCR-9732828. E-mail: *ladner@cs.washington.edu*.

1 Introduction

The *windows scheduling problem* is defined as follows with a terminology that is borrowed from a broadcasting system environment.

> Given are h slotted channels and n pages, $1, \ldots, n$, each is associated with a window $w_i \geq 1$ that is an integral number. Is it possible to schedule the n pages on the h channels, one page per one channel at each time slot, such that the gap between two consecutive appearances of page i is no more than w_i?

If for a given h and $W = \langle w_1, \ldots, w_n \rangle$ the answer to the windows scheduling problem is positive, the schedule that solves the problem is called a $\langle h, W \rangle$-schedule. The natural optimization problem is to find for a given W the minimum h, denoted by $H(W)$, such that there exists a $\langle h, W \rangle$-schedule. We call this problem the *optimal windows scheduling problem.* Since, the optimal windows scheduling problem is NP-Hard even for one channel [3], we will be looking for approximation solutions.

As an example of a windows scheduling problem consider the vector $W = \langle 2, 4, 5 \rangle$ which can be scheduled on one channel as follows:

$$\begin{bmatrix} 2 & 4 & 2 & 5 & 2 & 4 & 2 & 5 & \cdots \end{bmatrix}$$

Note that 2 is scheduled every 2 slots and 4 is scheduled every 4 slots, but 5 is scheduled every 4 slots. Clearly, this problem cannot be solved in any fewer number of channels so that $H(W) = 1$.

The *harmonic windows scheduling problem* is a special case of the windows scheduling problem in which $w_i = i$ for $1 \leq i \leq n$. If for given h and n the answer to the harmonic windows scheduling problem is positive, then the schedule that solves the problem is called a $\langle h, n \rangle$-schedule. Here, for a given n the minimum h such that there exists a $\langle h, n \rangle$-schedule is denoted by $H(n)$. However, an optimization objective that is more natural to the harmonic windows scheduling problem is to find for a given h the maximum n, denoted by $N(h)$, such that there exists a $\langle h, n \rangle$-schedule. We call this problem the *optimal harmonic windows scheduling problem.*

It is not difficult to see that $N(1) = 1$ and $N(2) = 3$ using the $\langle 2, 3 \rangle$-schedule

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 2 & 3 & 2 & 3 & \cdots \end{bmatrix}$$

A more interesting example is the $\langle 3, 9 \rangle$-schedule demonstrating that $N(3) \geq 9$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 2 & 4 & 2 & 5 & 2 & 4 & 2 & 5 & 2 & 4 & 2 & 5 & \cdots \\ 3 & 6 & 7 & 3 & 8 & 9 & 3 & 6 & 7 & 3 & 8 & 9 & \cdots \end{bmatrix}$$

This $\langle 3, 9\rangle$-schedule is known from the literature [15], but it is unknown whether the schedule is optimal. There may be a $\langle 3, 10\rangle$-schedule, but there is no $\langle 3, 11\rangle$-schedule. Note also that the windows scheduling problem $W = \langle 2, 4, 5\rangle$ has a solution as the second channel of the $\langle 3, 9\rangle$-schedule above. Thus, the windows scheduling problem is not only a generalization of the harmonic windows scheduling problem, but solutions to windows scheduling problems can be merged together to solve a harmonic windows scheduling problem.

We note that the performance criterion for these two optimization problems is based on the *max* metric and not the *average* metric. That is, the next appearance of a page depends only on its previous appearance. In the traditional scheduling problems the focus was on the average metric where in a way the next appearance of a page depends on all of its previous appearances.

The optimal windows scheduling problem is a formulation of the problem of minimizing the bandwidth of push systems that support guaranteed delay for the broadcasted pages. The optimal harmonic windows scheduling problem is an alternative formulation of the problem of minimizing the guaranteed startup delay in recently proposed media-on-demand systems.

1.1 The push systems application

In *push systems*, servers broadcast pages over broadcast channels to clients. Clients who wish to access one of the broadcasted pages listen to the channels until they receive this page. In traditional push systems (e.g., the Teletext problem [2] and the Broadcast Disks problem [1]), the server broadcasts more popular pages more frequently to minimize the average access time for clients. In some situations, the server does not have this freedom since it broadcasts information of some providers who "paid" for some quality of service.

These situations were considered by [7, 4]. In their push systems model they assume that the broadcasting environment is composed of three groups of "players": (i) Clients who wish to access information pages from broadcast channels. (ii) Servers who broadcast the information pages on channels. (iii) Providers who supply the information pages. Servers "sell" quality of service to providers who "pay" proportionally to the inverse of the maximum gap (access time) between any two appearances of their page on any of the channels.

After the selling process, the server is left with the optimal windows scheduling problem. That is, the server tries to minimize the number of channels that are needed to guarantee the quality of service "bought" by the providers.

1.2 The Media-on-Demand application

Media-on-Demand is the demand by clients to playback, view, listen, or read various types of media such as video, audio, or large files with as small as possible startup delays and with no interruptions. The solution of dedicating a private channel to each client for the required media is implausible even with the ever growing availability of network bandwidth. Thus, multicasting popular media to groups of clients seems to be the ultimate solution to the ever growing demand for media. The first, and most natural, idea to exploit the advantage of multicasting is to batch clients together. This implies a tradeoff between the overall server bandwidth and the guaranteed startup delay. The main advantage of the batching solutions lies in their simplicity. The main disadvantage is that the guaranteed startup delay may be still too large.

The pyramid broadcasting paradigm, pioneered by Viswanathan and Imielinski [17], was the first solution that reduced dramatically the bandwidth requirements for servers by using larger receiving bandwidth for clients and by adding buffers to clients. The novelty of the pyramid broadcasting paradigm is that clients with more complicated equipment are able to receive more bandwidth than they need for playback. This way they can buffer later portions of the required transmission to playback them on time. Many papers followed this line of research with various models. All of them demonstrated that with added complexity to the system, a huge improvement can be achieved over the traditional batching solutions.

To demonstrate the huge gain, suppose first that a server dedicates 2 channels to a 2 hour movie. In the traditional systems, by staggering the transmissions over the two channels, the server can guarantee a one hour startup delay. In systems with the new paradigm, the server can guarantee a startup delay of 40 minutes. Now suppose that a server dedicates 12 channels to a 2 hours movie. In this case, the improvement will be from 10 minutes to less than 2 seconds with a simple solution using the new paradigm and to less than one tenth of a second with a more complicated solution found in this paper.

In this paper, we adopt the model of [9, 11] that focuses on the tradeoff between the server bandwidth and the guaranteed startup delays and ignores the other two parameters the receiving bandwidth of clients and their buffer size. In this model, the transmission is divided into n equal size segments and the time is slotted with slots of size $1/n$ of the transmission length. Servers broadcast the n segments on h channels according to some transmission schedule. Clients who wish to receive the transmission wait for a beginning of a slot and thereafter they buffer the first appearance of segment i for $1 \leq i \leq n$. If for all $1 \leq i \leq n$, segment i appears in one of the ith slot after the client arrival time, then the client is guaranteed an uninterrupted playback. This is because when the client needs to playback the ith segment, it is either in its buffer or is transmitted on one of the channels. It is assumed that clients have large enough buffers that can store a full transmission and that they can receive data from all the channels concurrently.

We demonstrate this model with the simple case of a server with two channels. The transmission is divided into three segments. The first segment is broadcasted repeatedly on the first channel. The second and the third segments are broadcasted alternately on the second channel. A client has two possible starting times for receiving the transmission. The first is before a slot in which the second segment is broadcasted on the second channel and the second is before a slot in which the third segment is broadcasted on the second channel. (i) In the first time slot, in both cases, the client playbacks the first segment from the first channel. At the same time, in the first case, the client buffers segment two and in the second case the client buffers segment three. (ii) In the second time slot, in the first case the client playbacks segment two from its buffer while buffering segment three. In the second case, the client playbacks segment two from the second channel. (iii) In the third time slot, in both cases, the client playbacks segment three from its buffer. In this example, the guaranteed startup delay is one third of the transmission length.

A necessary and sufficient condition for clients to view the transmission with no interruptions is that segment i is broadcasted at least once in any consecutive i slots. In particular, a channel must be dedicated to the first segment and the second segment must appear at least every other slot on one of the other channels. Thus, if a $\langle h, n\rangle$-schedule exists, then clients can view the movie with no interruption with a guaranteed startup delay of $1/n$ of the movie length. To see this, assume first that clients arrive only at the beginning of slots, in this case a client may view the movie immediately with no interruptions. This is because the ith segment will be either on one of the channels i time slots after the arrival time of the client or in the client's buffer. As a result, clients who arrive in arbitrary times need to wait for a beginning of a slot in order to view the movie with no interruptions. The maximum startup delay is therefore $1/n$ of the movie length.

Remark: A "fractional" version of the problem allows mixing segments together in each channel. With this capability, the papers [10, 13, 14] present solutions that almost match the lower bound of [6]. However, this fractional model does not apply to the general windows scheduling problem and to the push systems application. Furthermore, even for the Media-on-Demand application, in many systems the "integral" version of the problem is more realistic.

1.3 Previous results and our contributions

To the best of our knowledge there are no published solutions for the optimal windows scheduling problem as formulated in this paper. Papers [8, 11] present a simple $\left\langle h, 2^h - 1\right\rangle$-schedule for the harmonic windows scheduling problem. The best known results for the harmonic windows scheduling problem appear in [15, 12]. The Pagoda scheme ([15]) is a $\langle 3, 9\rangle$-schedule and its generalization for $h > 3$. For an even $h > 3$, the Pagoda scheme is a $\left\langle h, (4/5)\sqrt{5}^h - 1\right\rangle$-schedule, and for an odd $h > 3$, it is a $\left\langle h, (2/\sqrt{5})\sqrt{5}^h - 1\right\rangle$-schedule. Asymptotically, the Pagoda scheme implies an $N(h) = O(\sqrt{5}^h) \approx O(2.236^h)$ solution. The new Pagoda scheme ([12]) deals with small values of h. Their lower bounds on $N(h)$ for $h = 3, 4, 5, 6, 7$ are $9, 26, 66, 172, 442$ respectively (see Table 1).

We first present an asymptotic result for the optimal windows scheduling problem. Define $h(W) = \sum_{i=1}^{n} 1/w_i$. A simple argument shows that $\lceil h(W) \rceil$ is a lower bound for $H(W)$ since page i requires at least $1/w_i$ portion of a channel. We show that a schedule exists that uses at most $h(W) + e \ln(h(W)) + 7.3595$[1] channels. In other words, this schedule is within a factor of $1 + O(\ln(h(W))/h(W))$ of optimal. This translates into a solution with $(e - O(\ln(h)/h))^h$ pages for the harmonic windows scheduling problem where h is the number of given channels and $0.561463e^h$ is an upper bound for $N(h)$.

The second part of our contribution focuses on "practical" approximate solutions to the optimal harmonic windows scheduling problem. Recall that in this case we fix the number of channels h and seek to find a $\langle h, n \rangle$-schedule for the largest possible n. An optimal solution would be a $\langle h, N(h) \rangle$-schedule. The asymptotic result implied by the solution to the optimal windows scheduling problem does not generate "good" enough schedules for small h. Most of the known constructions are based on more restrictive schedules. For example, the $\left\langle h, 2^h - 1 \right\rangle$-schedule (see Section 2), gives each page a fixed window size, schedules a page only on one channel, and on each channel schedules only consecutive pages. In this paper, all of the constructions schedule each page on only one channel. Most of them will give a page a fixed window size. We call such schedules *perfect schedules*.

We first describe a greedy schedule that approaches the upper bound for $N(h)$ for small values of h. This greedy scheme is based on a tree representation of perfect schedules introduced in [5]. The greedy strategy can be applied to the general windows scheduling problem as well, to obtain good bounds for $H(W)$ for cases in which $h(W)$ is small. Next we develop a technique that combines a $\langle h_1, n_1 \rangle$-schedule with a $\langle h_2, n_2 \rangle$-schedule to get a $\langle h_1 + h_2, ((n_1 + 1)(n_2 + 1))/2 - 1 \rangle$-schedule. With this technique, we get the best results for number of channels for which we cannot run the greedy algorithm. Finally, we present our non-trivial non-perfect schedules for $h = 4, 5, 6$ channels. These schedules were constructed using the tree representation technique with some modifications to accommodate non-perfect schedules.

# of channels	1	2	3	4	5	6	7	8	9	10	11	12
Upper bound	1	3	10	30	82	226	615	1673	4549	12366	33616	91379
Best bound	1	3	9	28	77	211	570	1573	4325	11759	31677	86428
Greedy	1	3	9	26	73	204	570	1573	4325	11759	31677	86428
Pagoda	1	3	9	26	66	172	442	499	1249	2499	6249	12499

Table 1: Results for 1 to 12 channels.

We summarize the bounds for 1 to 12 channels in Table 1. For a given h, we compare four values for $N(h)$: the upper bound, our best lower bound, the bound of the greedy algorithm,

[1] The symbol e is the base of the natural logarithm ln. All decimal constants are at the precision provided by the software package Mathematica [18].

and the best lower bound of the Pagoda scheme ([15]) and the New Pagoda scheme ([12]). To interpret Table 1 imagine that we can devote 8 channels to a 2 hour movie. Without any of these schemes a new movie can start every $120/8 = 15$ minutes. With the Pagoda scheme a movie can start every $7200/499 \approx 14.5$ seconds. With the greedy algorithm a movie can start every $7200/1573 \approx 4.6$ seconds. No such scheme can guarantee to start any faster than every $7200/1673 \approx 4.3$ seconds.

1.4 Paper organization

Section 2 presents some definitions and notation used later in the paper. Section 3 presents the upper bound and the asymptotic lower bounds for the optimal windows scheduling problem and the optimal harmonic windows scheduling problem. Section 4 describes the greedy algorithm which is an approximation algorithm for the optimal harmonic windows scheduling problem. Section 5 presents the combination technique for the optimal harmonic windows scheduling problem. Section 6 illustrates our best solutions for small numbers for the harmonic windows scheduling problem. Finally, Section 7 discusses some work in progress and open problems.

2 Preliminaries

In subsection 2.1, we present some of our definitions in a more formal way. In subsection 2.2, we describe a tree representation for perfect schedules that does not cover all the possible perfect schedules, but is enough to serve as a base for our greedy algorithm and to represent most of our solutions for small numbers. The idea is borrowed from [5]. We described what is needed to understand this paper.

2.1 Definitions and notation

A more formal definition of the windows scheduling problem and its special case the harmonic windows scheduling problem is as follows. We are asked to create a $h \times \infty$ matrix C in which the entry $C(j,t)$ is the page number that is scheduled at time slot t on the jth channel for $1 \leq j \leq h$ and $1 \leq t \leq \infty$. The requirement from the matrix is that for any $1 \leq i \leq n$ and $1 \leq t \leq \infty$, i is an element in the sub-matrix $C[1..h, t..t+w_i-1]$. That is, i appears in one of the entries in the following sub-matrix:

$$\begin{bmatrix} C(1,t) & \dots & C(1,t+w_i-1) \\ \vdots & & \vdots \\ C(h,t) & \dots & C(h,t+w_i-1) \end{bmatrix}$$

For the harmonic windows scheduling problem the same definition holds where $w_i = i$. Although a schedule is formally defined by an infinite matrix, any reasonable solution will be defined as the infinite concatenation of a finite schedule C which is a finite $h \times \ell$ matrix for some ℓ. Although it is not required, in all of our solutions each page is scheduled on a single channel. Such a solution has the property that for each page i, there is a channel h_i such that $i \notin \{C(h', t) : h' \neq h_i \text{ and } 1 \leq t \leq \ell\}$. Because of this we can focus on the schedules for individual channels.

Let $W = \langle w_1, \ldots, w_n \rangle$ be a window vector. For a sequence $S = \langle s_0, \ldots, s_{\ell-1} \rangle$ with $s_i \in \{1, \ldots, n\}$, define M_S to be the set of distinct pages in the sequence S. The sequence S is a potential row of a finite schedule for W. We define $S = \langle s_0, \ldots, s_{\ell-1} \rangle$ to be a *channel schedule* for W, if for each $i \in M_S$, if $i(1) < i(2) < \cdots < i(n_i)$ are all the indices such that $s_{i(j)} = i$ then $i(j+1) - i(j) \leq w_i$ for $1 \leq j < n_i$ and $\ell - i(n_i) + i(1) \leq w_i$. A channel schedule S is *perfect* if there exists a $w'_i \leq w_i$ such that $i(j+1) - i(j) = w'_i$ for $1 \leq j < n_i$ and $\ell - i(n_i) + i(1) = w'_i$. We call w'_i the window size of i in the perfect channel schedule S. The following proposition should be clear.

Proposition 1 *If S is a (perfect) channel schedule then so is the result of a finite concatenation of S with itself, a cyclic shift of S, and the reversal of S.*

A finite schedule for W can be formed from h channel schedules $S_1, \ldots S_h$ of lengths $\ell_1, \ldots, \ell_h$ respectively, where $\cup_{i=1}^{h} M_{S_i} = \{1, 2, \ldots, n\}$. Let ℓ be the least common multiple of $\{\ell_1, \ldots, \ell_h\}$. By a finite concatenation of each channel schedule with itself, S_i can be transformed into channel schedule of length ℓ. These channels schedules form the rows of a finite schedule matrix. A *perfect schedule* for W is one formed from perfect channel schedules. We define a perfect $\langle h, n \rangle$-schedule to be a perfect schedule for the window vector $\langle 1, 2, \ldots, n \rangle$ using h channels.

2.2 A tree representation for perfect channel schedules

Let T be an ordered tree with n labeled leaves. Assume first that all the labels (pages) are distinct. We describe a procedure that generates a perfect channel schedule on the n pages from T, computes the length of this channel schedule, and computes the window size of each page.

Procedure Tree-to-Schedule:

Input: A tree T with n leaves that are labeled with the labels $p_1, \ldots, p_n$ that represent the pages. Assume T has $d \geq 0$ subtrees.

Output: A perfect channel schedule S for these pages in which each page p_i has a fixed window size w'_i.

A leaf tree: If $n = 1$, then the schedule is $S = \langle p_1 \rangle$.

The recursive step: If $n > 1$ and therefore $d > 0$, do the following.

- Recursively construct the channel schedules $S_1, \ldots, S_d$ of all the d subtrees of T. Assume their respective lengths are $\ell_1, \ldots, \ell_d$.
- Replicate each channel schedule and make all of them to have the same length $\ell' = LCM\{\ell_1, \ldots, \ell_h\}$. Let the new schedules be $S'_1, \ldots, S'_d$. By Proposition 1, a replication of a perfect channel schedule results in a perfect channel schedule.
- The final channel schedule S of length $\ell = d\ell'$ is constructed by alternately picking ℓ' times the next page from the d channel schedules $S'_1, \ldots, S'_d$.

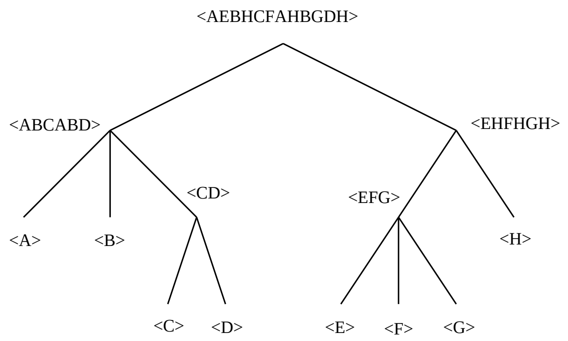

Figure 1: An example of a tree for pages A, B, C, D, E, F, G, H

A simple example is the round-robin tree. This is a tree whose root has d children all of them are leaves. If they are labeled by the numbers $p_1, p_2, \ldots, p_d$ then the channel schedule represented by this tree is the round-robin channel schedule $S = \langle p_1, p_2, \ldots, p_d\rangle$. A more complicated example is the tree of Figure 1. The labels are $A, B, \ldots, H$. Applying Procedure Tree-to-Schedule on the left and right subtrees yield the channel schedules $\langle ABCABD\rangle$ and $\langle EHFHGH\rangle$ respectively. Hence, the schedule representation of the tree is $\langle AEBHCFAHBGDH\rangle$.

We now give the condition for when Procedure Tree-to-Schedule generates a perfect channel schedule. Let $W = \langle w_1, \ldots, w_n\rangle$ be a window vector and let T be a tree with leaf labels $L \subseteq \{1, \ldots, n\}$. Let $W' = \cup_{i \in L}\{w_i\}$ be the window vector W restricted to the labels L. For $i \in L$, define $w'_i = \prod_{j=1}^{k} d_j$ where $d_1, \ldots, d_k$ are the degrees of all the ancestors of the leaf labeled i in T from the root to the parent of the leaf labeled i. That is, d_1 is the degree of the root and d_k is the degree of the parent of the leaf labeled i. Let $\ell = LCM\left\{w'_1, \ldots, w'_{|L|}\right\}$.

Theorem 2 *Procedure Tree-to-Schedule produces a perfect channel schedule for W' if and only if $w'_i \le w_i$ for all $i \in L$. In this case, w'_i is the window size of i in a channel schedule of length ℓ.*

Proof: We prove by induction on $|L|$ that the sequence $S = \langle s_0, \ldots, s_{\ell-1}\rangle$ produced by Procedure Tree-to-Schedule has the following *perfect* property:

for each $i \in L$, if $i(1) < i(2) < \cdots < i(n_i)$ are all the indices such that $s_{i(j)} = i$ then $i(j+1) - i(j) = w'_i$ for $1 \le j < n_i$ and $\ell - i(n_i) + i(1) = w'_i$.

The theorem follows immediately from this.

For trees with one leaf this is clearly true. If the root of T has d_1 children, then by the induction hypothesis the procedure run on each of the d_1 subtrees produces the sequences $S_1, \ldots, S_{d_1}$ each with the perfect property. It follows that $S'_1, \ldots, S'_{d_1}$ defined by the procedure also have the perfect property because each is just a replication of a sequence with the perfect property (Proposition 1). In the final schedule S produced by the Procedure, the label i occurs once every $w'_i = d_1 w''_i$ slots where $w''_i = \prod_{j=2}^{k} d_j$ and $d_2, \ldots, d_k$ are the degrees of ancestors of the leaf labeled i in the subtree of T that contains i. Hence, the sequence S satisfies the perfect property. Finally, it is not hard to verify from the definition of the procedure that the length of S is $\ell = LCM\left\{w'_1, \ldots, w'_{|L|}\right\}$. $\square$

Remark: So far we assumed that all the labels are distinct. This is not imperative. However, if two labels are the same, then the window size of the corresponding page may get more than one value. We use non-perfect trees in our solutions for $h = 4, 5, 6$.

Example: Figure 2 illustrates the h trees representing all the channel schedules in the $\left\langle h, 2^h - 1\right\rangle$ perfect schedule. The window size of $1 \le i \le 2^h - 1$ is 2^j where $2^j \le i < 2^{j+1}$.

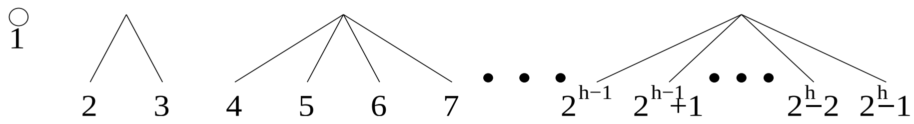

Figure 2: The $\left\langle h, 2^h - 1\right\rangle$-schedule.

3 Asymptotic bounds

Let $W = \langle w_1, w_2, \ldots, w_n\rangle$ be a window vector and let $h(W) = \sum_{i=1}^{n} 1/w_i$. Recall that $H(W)$ is the minimum h such that there exists a $\langle h, W\rangle$-schedule. In this section we show that $\lceil h(W)\rceil \le H(W) \le h(W) + e\ln(h(W)) + O(1)$. The lower bound comes from a simple observation, while the upper bound comes from the construction of a perfect schedule for W with the required bound on the number of channels. These bounds translate to bounds for $N(h)$ for the optimal harmonic windows problem.

3.1 Bounds for $H(W)$

We begin with the lower bound.

Theorem 3 *For any window vector $W = \langle w_1, \ldots, w_n \rangle$,*

$$H(W) \geq \lceil h(W) \rceil \ .$$

Proof: We need at least h channels where $h - 1 < \sum_{i=1}^{n} 1/w_i \leq h$. This is because page i requires at least a $1/w_i$ fraction of a channel. □

The upper bound on $H(W)$ is achieved by constructing a perfect schedule. There are a number of steps in the construction, we start with the simple case where the window sizes are all powers of 2. In this case there is a perfect schedule in which page i is scheduled exactly every w_i slots.

Lemma 4 *If all the w_i are of the form 2^{v_i} for some $v_i \geq 0$, then there exists a perfect schedule for the windows scheduling problem that uses exactly $H(W) = \lceil h(W) \rceil$ channels.*

Proof: Assume without loss of generality that $w_1 \leq \cdots \leq w_n$. If $h(W)$ is not an integer then choose a positive integer $y < 2^{v_n}$ such that $\lceil h(W) \rceil = h(W) + y/2^{v_n}$. Such a y exists since 2^{v_n} is divided by w_i for $1 \leq i \leq n$. In this case, we add y dummy pages each with a window of size 2^{v_n} and get that $\sum_{i=1}^{n+y} (1/w_i) = \lceil h(W) \rceil$. We then use the perfect schedule for the real and dummy pages in which the dummy pages are omitted. As a result, we may assume that $h(W)$ is a positive integer.

The proof is by induction on n. The base of the induction is when $w_1 = \cdots = w_n = 1$. Then $n = h(W)$ and the solution dedicates a separate channel to each page. Note also that the claim holds trivially for $n = 1$. Assume now that $w_n > 1$ for $n \geq 2$. We claim that $w_n = w_{n-1}$ since otherwise the sum cannot be an integral number because all the $1/w_i$ for $1 \leq i \leq n-1$ are strictly larger than $1/w_n$. Let W' be the vector of windows that is constructed from W by deleting w_{n-1} and w_n and adding $w' = w_n/2$. Since $w_{n-1} = w_n$, it follows that $h(W') = h(W)$. By the induction hypothesis on W', we get a solution with $h(W')$ channels in which page $1 \leq i \leq n-2$ is scheduled every w_i slots and the new page is scheduled every $w_n/2 = w_{n-1}/2$ slots. We replace this new page alternately with pages n and $n-1$. In this way these two pages are scheduled each every $w_n = w_{n-1}$ slots. □

As a consequence of Lemma 4, we have our first upper bound that is achieved by simply rounding the window sizes down to the nearest power of 2.

Lemma 5 *For any window vector W, there exists a perfect schedule that uses no more than $\lceil 2h(W) \rceil$ channels.*

Proof: Let $W = \langle w_1, w_2, \ldots, w_n \rangle$. For each w_i let v_i be the nonnegative integer such that $2^{v_i} \leq w_i < 2^{v_i+1}$. Let $W' = \langle 2^{v_1}, \ldots, 2^{v_n} \rangle$. Any schedule for W' is also a schedule for W. By Lemma 4, there is a perfect schedule for W' that uses $\lceil h(W') \rceil$ channels. Since $w_i/2 < 2^{v_i}$,

$$h(W') = \sum_{i=1}^{n} \frac{1}{2^{v_i}} < 2 \sum_{i=1}^{n} \frac{1}{w_i} = 2h(W).$$

Thus, at most $\lceil 2h(W) \rceil$ channels are used in the schedule. $\square$

We now improve the upper bound by generalizing Lemma 4. We consider instances in which all the window sizes are powers of 2 multiplied by the same number.

Lemma 6 *If all the w_i are of the form $u2^{v_i}$ for some $u \geq 1$ and some $v_i \geq 0$, then there exists a perfect schedule for the windows scheduling problem that uses exactly $H(W) = \lceil h(W) \rceil$ channels.*

Proof: Consider the set of windows $w'_i = w_i/u$. By Lemma 4, we can schedule the pages with the w'_i windows on

$$\left\lceil \sum_{i=1}^{n} \frac{1}{w'_i} \right\rceil = \left\lceil u \sum_{i=1}^{n} \frac{1}{w_i} \right\rceil \leq u \lceil h(W) \rceil \ .$$

channels. If the above is a strict inequality, we add dummy channels with empty schedules to get a schedule with exactly $u \lceil h(W) \rceil$ channels. We now combine sets of u channels together in a round robin fashion by picking pages alternately from the u channels. In this way the window of each page is multiplied by u. This is fine since by definition $w_i = uw'_i$. Overall we reduced the number of channels by a factor of u. Hence we used at most $\lceil h(W) \rceil$ channels. $\square$

Our goal is to construct an algorithm that for a given window vector W creates a perfect schedule with about $h(W) + e\ln(h(W))$ channels. The algorithm is recursive and is based on Lemmas 4 and 6. We first give a high level description of the algorithm with an illustrative example. The exact details of the algorithm and its analysis appear in the proof of Lemma 7.

An instance of the algorithm is determined by two parameters k and x that are optimized to obtain the best bound. First assume that $k = 1$. Here we use Lemma 5 to get a schedule with at most $\lceil 2h(W) \rceil$ channels. Now, fix $k > 1$. Partition the vector of windows into x vectors denoted by W_u for $x \leq u < 2x$. Each window $w_i \in W_u$ will be rounded down to a window w'_i such that $w'_i = 2^{v_i}u'$ where $u = 2^v u'$ for an odd number u'. We round down the window w_i such that w'_i is maximal. The set of all the windows w'_i such that $w_i \in W_u$ is denoted by W'_u. By Lemma 6, we need $\lceil h(W'_u) \rceil$ channels to schedule all the new windows in W'_u. A careful inspection of the proof of this lemma reveals that this schedule uses $\lfloor h(W'_u) \rfloor$ channels perfectly with no "waste" where the remaining windows (if exist) are scheduled on the last channel. This implies a partition of the set W into the perfect vector W^p and the residual vector W^r. Each window $w_i \in W^p$ is scheduled perfectly according to its corresponding w'_i. To schedule the windows in W^r, we recursively apply the algorithm for some $k' < k$.

For each k, we pick the best value for x. We will show that when x is larger then the w'_i values are closer to the original w_i values. However, the residual set W^r is too big. In order to use the recursion on a smaller residual set, we need a smaller value for x that in turn implies that some of the w'_i are too small compared with their corresponding w_i. This is the tradeoff that dictates the optimization process for the value of x. Finally, we will find the maximum value for k that can be used by this recursive algorithm to get our desired bound.

Example: Consider the following window vector:

$$W = \langle 2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19 \rangle \ .$$

It follows that $2 < h(W) < 3$ and therefore we need at least 3 channels to schedule the windows in W. Let examine first the case $k = 1$ (c.f. Lemma 5) in which the new set of windows is

$$W' = \langle 2,2,4,4,4,4,8,8,8,8,8,8,8,8,16,16,16,16 \rangle \ .$$

The best schedule for W' requires 4 channels. Now assume $k = 2$ and choose $x = 3$. We get the following three vectors W_3, W_4, W_5 and their corresponding three vectors W_3', W_4', W_5'.

$$\begin{aligned}
W_3 &= \langle 3,6,7,12,13,14,15 \rangle \\
W_3' &= \langle 3,6,6,12,12,12,12 \rangle \\
W_4 &= \langle 2,4,8,9,16,17,18,19 \rangle \\
W_4' &= \langle 2,4,8,8,16,16,16,16 \rangle \\
W_5 &= \langle 5,10,11 \rangle \\
W_5' &= \langle 5,10,10 \rangle
\end{aligned}$$

For example, the window 7 is rounded down to 6 and therefore belongs to W_3 and the windows $16, 17, 18, 19$ all are rounded down to 16 and therefore belong to W_4. We can use one channel to schedule all the windows in W_3'. We use another channel to schedule the windows $\langle 2,4,8,8 \rangle$ from W_4'. Hence, we get

$$\begin{aligned}
W^p &= \langle 2,3,4,6,7,8,9,12,13,14,15 \rangle \\
W^r &= \langle 5,10,11,16,17,18,19 \rangle
\end{aligned}$$

Now we apply the case $k = 1$ to schedule the windows in W^r. We get

$$W^{r\prime} = \langle 4,8,8,16,16,16,16 \rangle$$

that needs only one channel. All together, we need 3 channels: 2 for W^p and 1 for W^r.

We are now ready for the major technical lemma which leads to our upper bound. Define r to be a mapping from the positive integers to the reals by

$$r(k) = \begin{cases} 1 & \text{for } k = 1, \\ r(k-1) + \frac{e^{(k-1)/k}}{(e^{(k-1)^2/k} - 1)} & \text{for } k \geq 2. \end{cases}$$

Obviously, the function r is a monotonic increasing function. Furthermore, the $\lim_{k \to \infty} r(k)$ exists and is approximately 4.6412.

Lemma 7 *For any window vector W and all positive integers k, if $e^{k-1} < h(W) \leq e^k$, then there exists a perfect schedule with number of channels bounded above by*

$$h(W) + kh(W)^{1/k} + r(k) \ .$$

Proof: Let $W = \langle w_1, w_2, \ldots, w_n \rangle$. We begin with $k = 1$. By Lemma 5, for any value of $h(W)$, there exists a perfect schedule, denoted by S_1, that uses no more than $\lceil 2h(W) \rceil < 2h(W)+1 = h(W) + 1h(W)^{1/1} + r(1)$ channels.

Let $k > 1$. Let $e^{k-1} < h(W) \leq e^k$ and assume that the lemma holds for k', $1 \leq k' < k$, using the perfect schedule $S_{k'}$. The schedule S_k for W is defined as follows. Let $x = \left\lfloor h(W)^{(k-1)/k} \right\rfloor$. We partition the n windows of window vector W into the x window vectors $W_x, \ldots, W_{2x-1}$ and define a new window size $w'_i \leq w_i$ for each window w_i as follows:

Case $w_i < x$**:** Let $x \leq 2^{v_i} w_i = u < 2x$ for $v_i > 1$.
Put w_i in W_u and set $w'_i = w_i$.

Case $w_i \geq x$**:** Let $2^{v_i} u \leq w_i < 2^{v_i}(u+1)$ for $x \leq u < 2x$ and $v_i \geq 1$.
Put w_i in W_u and set $w'_i = 2^{v_i} u$.

We created x independent window vectors $W'_x, \ldots, W'_{2x-1}$ where W'_u contains those w'_i such that $w_i \in W_u$. All the windows in W'_u have the form $u'2^v$ for an odd u' such that $u = u'2^{v'}$. Hence by Lemma 6, there exists a perfect schedule T_u for W'_u with $\lceil h_u \rceil$ channels, where $h_u = h(W'_u) = \sum_{w'_i \in W'_u} 1/w'_i$. A careful inspection of the proof of this lemma reveals that this schedule uses $\lfloor h_u \rfloor$ channels perfectly with no "waste" where the remaining windows (if exist) are scheduled on the last channel. This implies a partition of the vector W_u into the perfect vector W^p_u and the residual vector W^r_u. W^p_u contains all the windows $w_i \in W_u$ for which their corresponding pages are scheduled on one of these no waste channels. W^r_u contains the rest of the windows in W_u. As a result, $h^p_u = \sum_{w_i \in W^p_u} 1/w'_i$ is an integer and $\sum_{w_i \in W^r_u} 1/w'_i < 1$. Since $1/w_i \leq 1/w'_i$, it follows that $h^r_u = \sum_{w_i \in W^r_u} 1/w_i < 1$. Define $W^p = \cup_{x \leq u < 2x} W^p_u$ and $W^r = \cup_{x \leq u < 2x} W^r_u$. The schedule S_k schedules all the windows in W^p using the perfect schedules T_u with exactly $\sum_{u=x}^{2x-1} h^p_u$ channels.

Before, describing how to schedule the pages whose windows are in W^r, we analyze the number of channels needed for W^p. For the windows scheduling problem W^p, window sizes $< 2x$ do not change and window sizes $\geq 2x$ are rounded down in each schedule T_u. The worst case happens when $w_i = 2^{v_i}(u+1) - 1$ for $u = x$ and therefore $w'_i = 2^{v_i} x$. This implies that $\frac{w_i}{w'_i} \leq \frac{x+1}{x}$. Hence, the number of channels in the schedule for W^p is

$$\begin{aligned}
\sum_{w_i \in W^p} \frac{1}{w'_i} &\leq \frac{x+1}{x} \sum_{w_i \in W^p} \frac{1}{w_i} = \frac{x+1}{x} h(W^p) = \frac{x+1}{x}(h(W) - h(W^r)) \\
&\leq h(W) + \frac{h(W)}{x} - h(W^r) \ . \qquad (1)
\end{aligned}$$

The quantity $h(W)/x$ can be bounded as follows. Since $x = \left\lfloor h(W)^{(k-1)/k} \right\rfloor$, $k > 1$, and $e^{k-1} < h(W)$, we have

$$h(W)/x \ = \ h(W)/\left\lfloor h(W)^{(k-1)/k} \right\rfloor$$

$$
\begin{aligned}
&\leq h(W)/(h(W)^{(k-1)/k}-1) \\
&= h(W)^{1/k}/(1-h(W)^{-(k-1)/k}) \\
&= h(W)^{1/k}\sum_{i=0}^{\infty}(h(W)^{-(k-1)/k})^i \\
&= h(W)^{1/k}+h(W)^{1/k}/(h(W)^{(k-1)/k}-1) \\
&< h(W)^{1/k}+e^{(k-1)/k}/(e^{(k-1)^2/k}-1) \qquad (2)
\end{aligned}
$$

The final inequality holds because the function $z/(z^{k-1}-1)$ is decreasing for all $k>1$ and $z>1$.

We are left with the remaining windows, W^r, to be scheduled. Because $h_u^r<1$ for each u and $h(W)\leq e^k$, we have $h(W^r)\leq e^{k-1}$ as can be seen in the derivation

$$
\begin{aligned}
h(W^r) &= \sum_{w_i\in W^r}\frac{1}{w_i}=\sum_{u=x}^{2x-1}h_u^r\leq x=\left\lfloor h(W)^{(k-1)/k}\right\rfloor\leq h(W)^{(k-1)/k} \qquad (3)\\
&\leq (e^k)^{(k-1)/k}=e^{k-1}\ . \qquad (4)
\end{aligned}
$$

There are two cases to consider depending on whether $h(W^r)\leq 1$.

Case 1: $h(W^r)\leq 1$. In this case by Lemma 5, W^r has a perfect schedule with number of channels bounded above by $\lceil 2h(W^r)\rceil\leq 2h(W^r)+1$. By this and inequalities (1) and (2), there is a perfect schedule for W that has number of channels bounded above by

$$
h(W)+h(W)^{1/k}+h(W^r)+1+e^{(k-1)/k}/(e^{(k-1)^2/k}-1)\ .
$$

Since $k-1\geq 1$ and $h(W)>e^{k-1}$ it follows that $1<(k-1)h(W)^{1/k}$. Since $h(W^r)\leq 1$ it follows that $h(W^r)<(k-1)h(W)^{1/k}$. Furthermore, $1+e^{(k-1)/k}/(e^{(k-1)^2/k}-1)\leq r(k)$. Hence, we have our bound.

Case 2: $h(W^r)>1$. By inequality (4), we have $e^{k'-1}<h(W^r)\leq e^{k'}$ for some $k'<k$. By the induction hypothesis, the perfect schedule $S_{k'}$ for W^r has number of channels bounded by

$$
h(W^r)+k'h(W^r)^{1/k'}+r(k')\ .
$$

Claim: $k'h(W^r)^{1/k'}\leq (k-1)h(W^r)^{1/(k-1)}$.

Proof of Claim: We show that $az^{1/a}\leq bz^{1/b}$ for $1\leq a\leq b$ for $1\leq z\leq e^a$. When $z=1$ we have $az^{1/a}\leq bz^{1/b}$ and as z goes to infinity $bz^{1/b}$ is smaller than $az^{1/a}$. The functions $az^{1/a}$ and $bz^{1/b}$ intersect exactly once because the latter function is more concave than the former. It therefore suffices to show that $az^{1/a}\leq bz^{1/b}$ for $z=e^a$. By calculus the function e^y/y in the range $(0,\infty)$ has a minimum at $y=1$. Hence, $e\leq e^y/y$ for all $y>0$. Letting $y=a/b$ we have $e\leq e^{a/b}/(a/b)$ which is equivalent to $a(e^a)^{1/a}\leq b(e^a)^{1/b}$. The claim follows for $a=k'\leq b=k-1$ since $h(W^r)\leq e^{k'}$.

Because of the above claim and the fact that $r(m)$ is increasing in m, we have the number of channels for the schedule $S_{k'}$ for W^r is bounded by

$$h(W^r) + (k-1)h(W^r)^{1/(k-1)} + r(k-1) \ .$$

By inequality (3), we have the number of channels for the schedule $S_{k'}$ for W^r is bounded by

$$h(W^r) + (k-1)h(W)^{1/k} + r(k-1) \ . \tag{5}$$

Combining the bounds (1), (2), and (5), we have the number of channels for the combined perfect schedules for W^p and W^r together is bounded by

$$h(W) + kh(W)^{1/k} + r(k-1) + e^{(k-1)/k}/(e^{(k-1)^2/k} - 1) \ ,$$

which by the definition of r yields our result. □

We are now ready to prove the upper bound on $H(W)$.

Theorem 8 *Every window vector W, with $h(W) > 1$, has a perfect schedule using number of channels bounded above by $h(W) + e\ln(h(W)) + \eta$, where $\eta \approx 7.3595$.*

Proof: The quantity $\eta = e + \lim_{k\to\infty} r(k)$. Choose $k = \lceil \ln(h(W)) \rceil$. Hence, $e^{k-1} < h(W) \leq e^k$. By Lemma 7, there exists a perfect schedule with number channels bounded above by $h(W) + kh(W)^{1/k} + r(k)$. This is no more than $h(W) + (\ln(h(W)) + 1)(e^k)^{1/k} + r(k)$ which is equivalent to the claim of the theorem. □

The proof of Theorem 8 implies an approximation algorithm for the optimal windows scheduling problem.

Theorem 9 *There exists an algorithm for the optimal windows scheduling problem yielding a solution that is within a factor of $1 + O(\ln(h(W))/h(W))$ of the optimal solution for any window vector W.*

Proof: Let the algorithm of Theorem 8 produce a schedule with $A(W)$ channels. Recall that $H(W)$ is the optimal number of channels. Without loss of generality we can assume that $h(W) \geq 2$. We have

$$h(W) \leq H(W) \leq A(W) \leq h(W) + e\ln(h(W)) + \eta \ .$$

Hence,

$$\frac{A(W)}{H(W)} \leq \frac{h(W) + e\ln(h(W)) + \eta}{h(W)} = 1 + O\left(\frac{\ln(h(W))}{h(W)}\right) .$$

□

3.2 Bounds on $N(h)$

In the optimal harmonic windows scheduling problem the window vectors are restricted to be of the form $W_n = \langle 1, 2, \ldots, n \rangle$. Our lower bound on $H(W_n) = H(n)$ (Theorem 3) leads to the following natural upper bound on $N(h)$.

Theorem 10

$$N(h) \le c \cdot e^h$$

for $c = e^{-0.57721} \approx 0.56147$.

Proof: By Theorem 3, $H(n) \ge h(W_n) = \sum_{i=1}^{n} 1/n$. The quantities $\sum_{i=1}^{n} 1/n$ $(n \ge 1)$ are the well known harmonic numbers which are known to be bounded below by $\ln(n) + \gamma$ where $\gamma \approx 0.57721...$ is Euler's constant. Hence, $h = H(N(h)) > \ln(N(h)) + \gamma$, which implies the theorem. $\square$

We conclude this section with a lower bound for $N(h)$ that is derived from the upper bound on $H(W)$ from Theorem 8.

Theorem 11

$$N(h) \ge \frac{e^h}{e^{\eta+\gamma} h^e} \approx \frac{e^h}{e^{7.93669} h^e} \ .$$

Proof: By Theorem 8, we are looking for n such that $h(W_n) + e\ln(h(W_n)) + \eta \le h$. For any such n, $N(h) \ge n$. Since $h(n) = \ln(n) + \gamma$, it follows that for $n = \left\lfloor \frac{e^h}{e^{\eta+\gamma} h^e} \right\rfloor$,

$$h(W_n) = h\left(\left\lfloor \frac{e^h}{e^{\eta+\gamma} h^e} \right\rfloor\right) = \ln\left(\left\lfloor \frac{e^h}{e^{\eta+\gamma} h^e} \right\rfloor\right) + \gamma \le h - e\ln(h) - \eta \ .$$

Hence,

$$h(W_n) + e\ln(h(W_n)) + \eta \le h - e\ln(h) - \eta + e\ln(h) + \eta = h \ .$$

. $\square$

With some arithmetic manipulations, one can prove the following corollary.

Corollary 12 *For large enough value of* h *we have that* $N(h) \ge \left(e - \frac{c\ln(h)}{h}\right)^h$ *for some constant* c.

Later in Section 5 we will see that the Pagoda scheme [15] and our combination technique generalization yield asymptotic results in which $N(h) \ge (e - c')^h$ for some constant c'. Thus, obviously Corollary 12 presents a better asymptotic result.

4 The greedy algorithm

In this section, we describe a greedy strategy for the harmonic windows scheduling problem. It will be clear how this strategy can be generalized to the general windows scheduling problem. However, we do not have analytical bounds for this greedy approach and we just use it as a heuristic to get better bounds than those produced by our other methods for the harmonic scheduling problem.

The greedy algorithm proceeds in rounds, where in the rth round a perfect placement for page r is found in the channel that minimizes the difference $r - r'$ where r' is the distance between placements of page r. The best placement would be to choose, if possible, the channel in which $r' = r$. This might not be possible, for example when there are just two channels and we have already perfectly placed 1 in the first channel and 2 in the second channel, then 3 must be placed in the second channel with a distance of 2 apart. In this case $r - r' = 1$. In order to keep track of where placements can be made we represent each channel by a tree, where pages are assigned only to some of the leaves of the tree. If in some round r, page r cannot be placed perfectly in any tree then the algorithm terminates having successfully placed $r - 1$ pages.

More formally, let an *open tree* be a tree whose leaves are labeled with two types of labels: pages and windows. A *closed tree* is an open tree that has only page labels. An *open forest* is a collection of open trees and a *closed forest* is a collection of closed trees. The algorithm maintains an open forest composed of h trees. Initially, all the trees are singleton open trees with window labels whose value is 1. The algorithm terminates when the forest becomes a closed forest. The algorithm runs in rounds. In round r, the algorithm places page r in one of the open trees as follows:

1. Let $w_1 \leq w_2 \leq \cdots \leq w_k$ be the ordered list of the labels of all the leaves in the forest whose labels are of type window.
   (a) Let $m_j = (r \bmod w_j)$ for $1 \leq j \leq k$.
   (b) Let i be the the maximal index for which m_i is the minimum among all the m_j.
   (c) Let $d_i = \lfloor r/w_i \rfloor$.
   (d) Let T_s be the tree that contains w_i.
2. If $d_i = 1$, then replace the window label w_i with the page label r in the tree T_s. We call this the *replacement* operation.
3. Otherwise, add d_i children to the leaf associated with w_i replacing this leaf with d_i new leaves. The first child is labeled with the page label r and the rest are labeled with the window label $w_i \cdot d_i$. We call this the *split* operation.

Note that since $m_i = r - w_i d_i$, we see that our choice of m_i achieves the goal of minimizing $r - r'$ where $r' = w_i d_i$ is the distance between placements of r in the perfect channel schedule that is produced by T_s.

Theorem 13 *The greedy algorithm constructs a perfect $\langle h, n\rangle$-schedule for some value n.*

Proof: By definition, the algorithm produces h trees that represent h perfect schedules. The value of n is the last round r in which the forest was still an open forest. It remains to show that indeed the window size of page i is no larger than i. This is true since in both the replacement operation and the split operation, the window size assigned to page i is $w_i d_i$ which is less than r by definition (step 1c). □

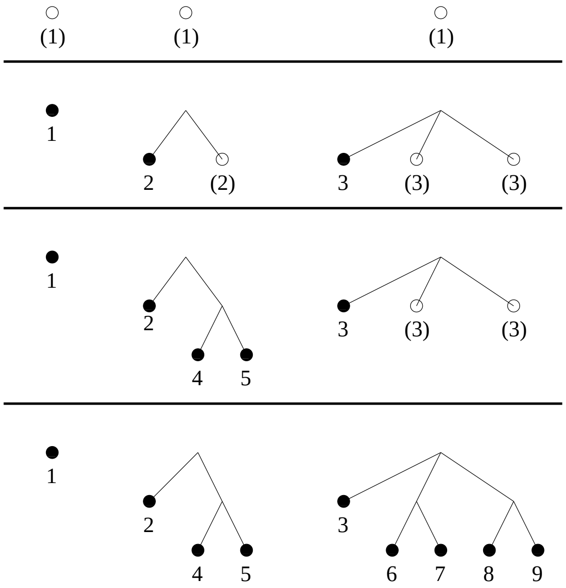

Figure 3: The evolution of the greedy algorithm for $h = 3$.

In Figure 3, we show the evolution of the greedy algorithm for $h = 3$. The final schedule is the perfect $\langle 3, 9\rangle$-schedule that first appeared in [15]. In the illustration, we use numbers for page labels and numbers in parenthesis for window labels. In addition, we use black circles and white circles to denote leaves with page labels and leaves with window labels respectively. At the top level, we illustrate the initial open forest that is composed of three singleton open trees whose labels are windows of size 1. In round 1, the label of the first tree becomes the page label 1. This tree is now a closed tree. In round 2, a split operation occurs in the second tree and it becomes a tree whose root has two children, one with a page label 2 and one with a window label 2. In round 3, a split operation occurs in the third tree and it becomes a tree whose root has three children one with a page label 3 and two with a window label 3. Note that since $(3 \bmod 1) < (3 \bmod 2)$, the algorithm selects for the split operation the window label 1 in the third tree rather than the window label 2 in the second tree. The second level of the figure illustrates the three trees after the third round. In round 4, the algorithm splits the node in the second tree with the window label 2 and creates two new leaves one with a page label 4 and one with a window label 4. This is because $(4 \bmod 2) < (4 \bmod 3)$. Hence, before round 5, the list of available windows size are $3, 3, 4$. Therefore, the algorithm assigns the page label 5 to the leaf in the second tree with the window label 4. Hereafter, the second tree is closed. The third level of the figure illustrates the three trees after the fifth round. In round 6, the second child of the root of the third tree is split into two leaves one gets the page label 6 in round 6 and the other gets the page label 7 in round 7. Finally, in round 8, the third child of the root of the third tree is split into two leaves one gets the page label 8 in round 8 and the other gets the page label 9 in round 9. At this stage all the trees are closed and the algorithm terminates with the final round $r = 9$ and therefore $n = 9$ as well. The output of the algorithm is illustrated in the fourth level of the figure. Note, that pages $2, 3, 4, 6$ get the minimum possible window size whereas pages $5, 7, 8, 9$ get a smaller than required window size. However, this is the best known solution to the harmonic window scheduling problem for $h = 3$.

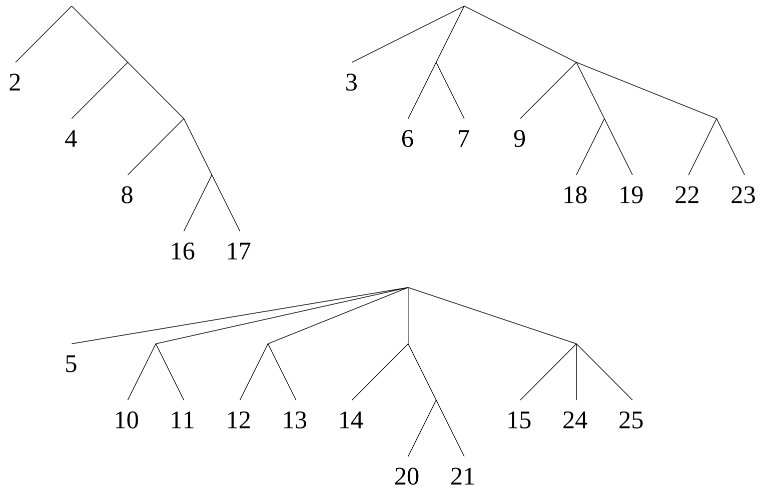

Figure 4: The greedy $\langle 4, 25\rangle$-schedule.

Figure 4 illustrates the output of the greedy algorithm for $h = 4$. The greedy outputs a $\langle 4, 25\rangle$ schedule where there exists a $\langle 4, 26\rangle$-schedule ([12]). In particular, this version of the greedy algorithm is not efficient for pages 24 and 25. It could do better by assigning 16 and 17 as siblings of 15 and then assigning 24, 25, and 26 as the children of the sibling of 8.

This leads us to modify the greedy algorithm. We propose two possible modifications. Both try to leave leaves with small window labels open as long as possible.

- The first modification changes the split operation. When d_i is a composite number, it is done in several steps following an increasing order of the prime factors of d_i. For example, if $d_i = 12$ for a node with a window label w_i. Then the node is twice split into two nodes and once into three nodes. Thus, creating five new leaves whose window labels are $2w_i, 4w_i, 12w_i, 12w_i, 12w_i$. One of the window label $12w_i$ becomes a page label but the rest remain window labels. In the greedy algorithm that was described earlier, there will be 11 new window labels $12w_i$ and one page label $12w_i$.

- In the second modification, the algorithm follows a different rule in step (1b) of the algorithm. It sometimes biases towards assigning the new label r to a large window label on the expense of not minimizing $r - r'$. This way, it leaves in the tree smaller window labels for possibly better split operations.

We implemented the greedy algorithm with and without the above two modifications. Unfortunately, it was not the case that one version always outperforms the other versions.

Although, in most of the cases the basic greedy algorithm with both modification was the best. We were able to run our implementations of the various greedy algorithm for $1 \leq h \leq 20$. The results of the best greedy algorithm for $1 \leq h \leq 12$ appear in Table 1. We would like to note that for $h \geq 7$ the greedy algorithm produces our best results and that for $15 \leq h \leq 20$ the greedy solution is within 3% of optimal.

5 The combination technique

In this section, we develop a technique for solving the optimal harmonic windows scheduling problem. We generalize the technique of [15]. Their Pagoda scheme is a special case of the results of this section. We show how to combine schedules together to get new schedules for larger number of channels. For example, we can combine the $\langle 3, 9\rangle$-schedule with the $\langle 4, 28\rangle$-schedule to produce a $\langle 7, 289\rangle$-schedule (see Theorem 15). However, for $h \leq 20$, the greedy algorithm outperforms any possible combination. Since we have the output of greedy only for $h \leq 20$, the combination technique yields our best results for $21 \leq h < H$ for some large integer H. An estimate for H is 1789 and is computed at the end of this section. For $h \geq H$ the asymptotic algorithm of Section 3 yields the best bound.

We start with a technical lemma. For $u \leq v$, let a $\langle h, u, v\rangle$-schedule be a schedule on the pages $u, \ldots, v$ on h channels such that page i appears at least once in any consecutive i slots for $u \leq i \leq v$. In particular, a $\langle h, n\rangle$-schedule is a $\langle h, 1, n\rangle$-schedule and a $\langle h-1, 2, n\rangle$ schedule since page 1 gets a dedicated channel. The techniques of this section show how to build new schedules from old ones. As such we define the following notation:

$$\langle h_1, u_1, v_1\rangle, \ldots, \langle h_m, u_m, v_m\rangle \Rightarrow \langle h, u, v\rangle,$$

if a $\langle h, u, v\rangle$-schedule can be constructed from $\langle h_1, u_1, v_1\rangle$- , ..., $\langle h_m, u_m, v_m\rangle$-schedules. The next lemma shows how to transform a $\langle h, u, v\rangle$ schedule into another schedule for pages from a larger range. We call this the Magnification Lemma.

Lemma 14 (Magnification Lemma) $\langle h, u, v\rangle \Rightarrow \langle h, \ell u, \ell(v+1)-1\rangle$ *for any integer* $\ell \geq 1$.

Proof: Let S be a $\langle h, u, v\rangle$-schedule. We transform S which is a schedule on the pages $u, \ldots, v$ into another schedule S' on the pages $\ell u, \ldots, \ell(v+1)-1$. We replace page u with pages $\ell u, \ldots, \ell(u+1)-1$ in a round-robin fashion. That is, the first appearance of u is replaced by ℓu, the second is replaced by $\ell u+1$, until the ℓth appearance of u that is replaced by $\ell(u+1)-1$. Then again the next appearance of u is replaced by ℓu and so on. Since in S the window size of u is at most u, it follows that each of the pages $\ell u, \ldots, \ell(u+1)-1$ is guaranteed to have a window of size ℓu. Thus the scheduling of these pages is as required. In general, in a similar round-robin fashion, we replace the appearances of page $u+j$ in S with the pages $\ell(u+j), \ldots, \ell(u+j+1)-1$ for $0 \leq j \leq v-u$. Since in S the window size of $u+j$ is at most $u+j$, it follows that each of the pages $\ell(u+j), \ldots, \ell(u+j+1)-1$ is guaranteed to have a window of size $\ell(u+j)$. Thus the scheduling of these pages is as required. We get that the largest page that replaces an appearance of v is $\ell(u+(v-u+1))-1=\ell(v+1)-1$. $\square$

As an example of Lemma 14, we magnify the $\langle 2,1,3\rangle$-schedule

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 2 & 3 & 2 & 3 & \cdots \end{bmatrix}$$

Choosing $\ell=10$, we can create the $\langle 2,10,39\rangle$-schedule

$$\begin{bmatrix} 10 & \cdots & 19 & 10 & \cdots & 19 & 10 & \cdots & 19 & 10 & \cdots & 19 & \cdots \\ 20 & \cdots & 29 & 30 & \cdots & 39 & 20 & \cdots & 29 & 30 & \cdots & 39 & \cdots \end{bmatrix}$$

The above lemma implies our main combination theorem. Here we use $\langle h,n\rangle$ as an abbreviation for $\langle h,1,n\rangle$.

Theorem 15 (Combination Theorem)

$$\langle h_1,n_1\rangle\,,\langle h_2,n_2\rangle\Rightarrow\langle h_1+h_2,(n_1+1)(n_2+1)-1\rangle\,.$$

Proof: The $\langle h_2,n_2\rangle$-schedule is by definition a $\langle h_2,1,n_2\rangle$-schedule. By Lemma 14, for $\ell=n_1+1$, there exists a $\langle h_2,n_1+1,(n_1+1)(n_2+1)-1\rangle$-schedule. The schedule with h_1+h_2 channels is obtained by scheduling pages $1,\ldots,n_1$ on the first h_1 channels following the $\langle h_1,n_1\rangle$-schedule and scheduling pages $n_1+1,\ldots,(n_1+1)(n_2+1)-1$ on the last h_2 channels. $\square$

From our examples in the introduction of $\langle 3,9\rangle$ and $\langle 2,3\rangle$ we can construct a $\langle 5,39\rangle$. The first three channels consists of the $\langle 3,1,9\rangle$-schedule and the last two channels consists of the $\langle 2,10,39\rangle$-schedule constructed above as an example of Lemma 14.

Corollary 16 $\langle h,n\rangle\Rightarrow\langle h+1,2n+1\rangle$.

Proof: Apply Theorem 15 using $\langle 1,1\rangle$ and $\langle h,n\rangle$. $\square$

The above corollary is simple, but serves as the base in constructing the folklore $\left\langle h,2^h-1\right\rangle$-schedule. This schedule is obtained by applying $h-1$ times Corollary 16 starting with the $\langle 1,1\rangle$-schedule. A better asymptotic result than the $\left\langle h,2^h-1\right\rangle$-schedule may be obtained by taking a known schedule on $h>1$ channels and applying the combination theorem, Theorem 15, again and again as in the next theorem.

Theorem 17 $\langle k,n\rangle\Rightarrow\left\langle \ell k,(n+1)^\ell-1\right\rangle$ *for any integer* $\ell\geq 1$.

Proof: We prove the theorem by induction on ℓ. For $\ell=1$ the theorem follows trivially. For $\ell>1$, assume the claim holds for $\ell-1$. That is, there exists a $\left\langle(\ell-1)k,(n+1)^{\ell-1}-1\right\rangle$-schedule. Then by Theorem 15, using this schedule and the $\langle k,n\rangle$-schedule there exists a $\left\langle(\ell-1)k+k,((n+1)^{\ell-1}-1+1)(n+1)-1\right\rangle=\left\langle \ell k,(n+1)^\ell-1\right\rangle$ schedule. $\square$

The following corollary is equivalent to Theorem 17.

Corollary 18 $\langle k,n\rangle \Rightarrow \left\langle h, \left(\sqrt[k]{n+1}\right)^h - 1\right\rangle$, *for* $h = \ell k$.

Corollary 18 implies the existence of a $\left\langle h, \left(\sqrt[3]{10}\right)^h\right\rangle$-schedule provided that 3 divides h and the existence of a $\left\langle h, \left(\sqrt[4]{29}\right)^h\right\rangle$-schedule provided that 4 divides h. The former is based on the $\langle 3,9\rangle$-schedule and the latter on the $\langle 4,28\rangle$-schedule. Both are better than the $\left\langle h, 2^h - 1\right\rangle$-schedule but asymptotically far from a potential $\left\langle h, 0.56147e^h\right\rangle$-schedule that is implied by Theorem 10.

A slightly better asymptotic result than the one in Theorem 17 can be obtained by using the fact that a $\langle h,n\rangle$-schedule is also a $\langle h-1,2,n\rangle$-schedule.

Theorem 19 $\langle k,n\rangle \Rightarrow \left\langle \ell(k-1)+1, 2((n+1)/2)^\ell - 1\right\rangle$ *for any* $\ell \geq 1$.

Proof: We prove the theorem by induction on ℓ. For $\ell = 1$ the theorem follows since $\ell(k-1)+1 = k$ and $2((n+1)/2)^\ell - 1 = n$. For $\ell > 1$, assume the claim holds for $\ell - 1$. That is, there exists a $\left\langle (\ell-1)(k-1)+1, 2((n+1)/2)^{\ell-1} - 1\right\rangle$-schedule. We have $\langle k,n\rangle \Rightarrow \langle k-1,2,n\rangle$-schedule by deleting the first channel. By Lemma 14 using the factor $((n+1)/2)^{\ell-1}$, we have $\langle k-1,2,n\rangle \Rightarrow \left\langle k-1, 2((n+1)/2)^{\ell-1}, 2((n+1)/2)^\ell - 1\right\rangle$. The claim of the theorem follows by adding the $k-1$ channels of the $\left\langle k-1, 2((n+1)/2)^{\ell-1}, 2((n+1)/2)^\ell - 1\right\rangle$-schedule on the pages $2((n+1)/2)^{\ell-1}, \ldots, 2((n+1)/2)^\ell - 1$ to the $(\ell-1)(k-1)+1$ channels of the $\left\langle (\ell-1)(k-1)+1, 2((n+1)/2)^{\ell-1} - 1\right\rangle$-schedule on pages $1, \ldots, 2((n+1)/2)^{\ell-1} - 1$ to obtain a $\left\langle \ell(k-1)+1, 2((n+1)/2)^\ell - 1\right\rangle$-schedule. $\square$

The following corollary is equivalent to Theorem 19.

Corollary 20 $\langle k,n\rangle \Rightarrow \left\langle h, 2\left(\sqrt[k-1]{(n+1)/2}\right)^{h-1} - 1\right\rangle$, *for* $h = \ell(k-1)+1$. *Equivalently,* $\langle k,n\rangle \Rightarrow \left\langle h, c\left(\sqrt[k-1]{(n+1)/2}\right)^{h} - 1\right\rangle$, *for* $h = \ell(k-1)+1$ *and* $c = 2/\sqrt[k-1]{(n+1)/2}$.

By Corollary 20, the $\langle 3,9\rangle$-schedule implies an approximately $\left\langle h, 0.894227\,(2.23607)^h\right\rangle$-schedule provided that 2 divides $h-1$ (the Pagoda scheme of [15]) and the $\langle 4,27\rangle$-schedule implies an approximately $\left\langle h, 0.829287\,(2.41014)^h\right\rangle$-schedule provided that 3 divides $h-1$. Note that we cannot use the $\langle 4,28\rangle$-schedule since in this case n is even.

Comparing the asymptotic results: We implemented the greedy algorithm for $1 \leq h \leq 20$ yielding in particular for $h = 20$ a $\langle 20, 265133522\rangle$-schedule. This in turn implies an asymptotic $\left\langle h, 0.747366(2.67606)^h\right\rangle$-schedule provided that 19 divides $h-1$ (Corollary 20). This value is "close" to the upper bound of $0.56147e^h$ on $N(h)$ (Theorem 10). Using Mathematica [18], we get that this schedule is smaller than the $e^h/(e^{7.93669}h^e)$ bound (Theorem 11) for $h > 1789$. That is, the combination technique yields our best results for h in the range [21..1789]. However, implementing the greedy algorithm for $h > 20$ would increase the 1789 threshold. On the other hand, implementing the asymptotic algorithm for $W = \langle 1, \ldots, n \rangle$ would definitely imply a better bound than the one from Theorem 11 and hence would decrease the 1789 threshold. This is because the bounds in Section 3 are mainly for the general windows scheduling problem.

6 Solutions for small h

In this section we describe our best constructions to the harmonic windows scheduling problem for small h. For $h = 2$ the general $\left\langle h, 2^h - 1 \right\rangle$ schedule shown in Figure 2 yields a $\langle 2, 3 \rangle$ schedule which is optimal. For $h = 3$ the greedy algorithm example shown in Figure 3 yields a $\langle 3, 9 \rangle$-schedule which is not known to be optimal. The best possible would be a $\langle 3, 10 \rangle$-schedule. For $h = 4, 5$, we illustrate the $\langle 4, 28 \rangle$ and the $\langle 5, 77 \rangle$ non-perfect schedules using the tree representation. We omit the details of our tree representation for the $\langle 6, 211 \rangle$ non-perfect schedule.

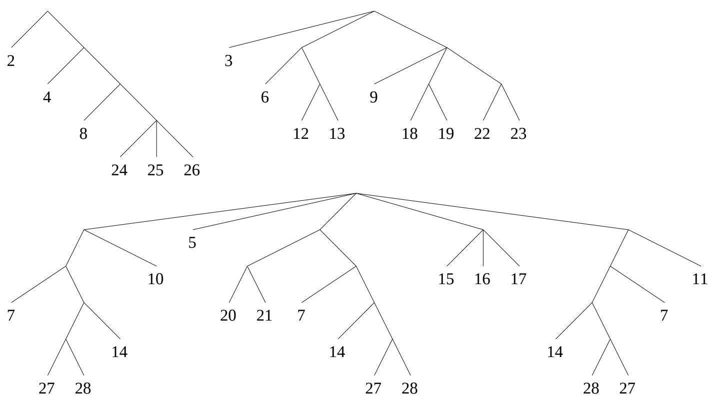

Figure 5: The $\langle 4, 28 \rangle$-schedule.

Our $\langle 4, 28 \rangle$-schedule is not a perfect schedule. The first channel is dedicated to page 1. Figure 5 uses the tree representation to illustrate the schedules of the second, the third, and the fourth channels. Channels 1,2, and 3 have perfect channel schedules but channel 4 does not. This is because pages $7, 14, 27, 28$ occur at multiple leaves of tree 4. The window sizes for page 7 are $7, 7, 6$ repeatedly, for page 14 they are $14, 13, 13$ repeatedly, and for pages $27, 28$ they are $27, 27, 26$ repeatedly.

Figure 6 illustrates the four trees representing the second, the third, the fourth, and the fifth schedules in the non-perfect $\langle 5, 77 \rangle$-schedule. The first two trees represent perfect schedules in the other two trees pages $11, 13, 32, 33, 34$ do not have a fixed window size.

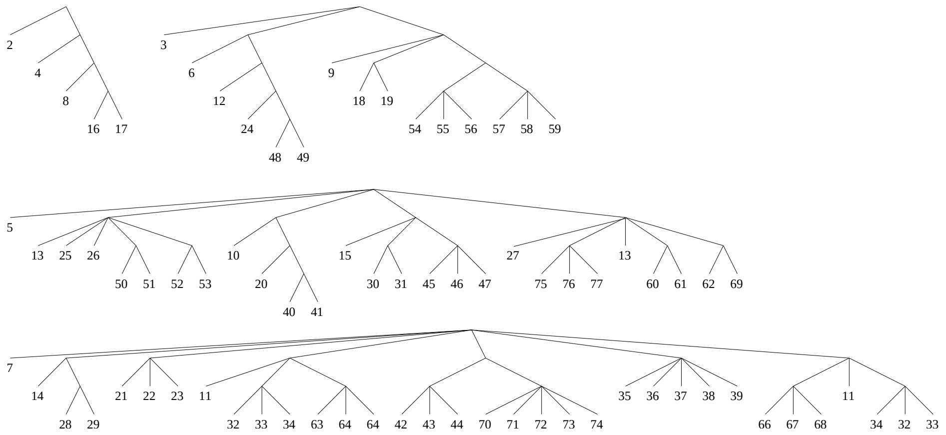

Figure 6: The $\langle 5, 77\rangle$-schedule.

7 Open problems

In this section, we discuss some open problems related to the windows scheduling problem, to the harmonic windows scheduling problem, and to the two motivating applications the push systems with guaranteed delay and the new media-on-demand systems.

The harmonic windows scheduling problem:

- Is the $\langle 3, 9\rangle$-schedule optimal? We conjecture that the answer is yes. However, to show this, more complicated arguments are needed to improve the upper bound of $N(3) \leq 10$ to be $N(3) \leq 9$. These arguments should rely on the *gcd* relationship among the numbers 2 to 9. However, we are able to show that for perfect schedules $N(3) \leq 9$.

- For the media-on-demand application efficient solutions for small values of h are important. Are there better schedules for $h = 4, 5, 6$? The gap for $h = 4$ is $28 \leq N(4) \leq 30$, for $h = 5$ it is $77 \leq N(5) \leq 82$, and for $h = 6$ it is $211 \leq N(5) \leq 226$. For $7 \leq h \leq 20$, our best solutions are outputs of the greedy algorithm. Is there a better algorithm?

- Is there any analytic bound for our greedy algorithm?

- The combination technique yields our best results for $21 \leq h \leq 1789$. A further improvement would be obtained by looking for the best combination for any h using a dynamic programming implementation of the Combination Theorem.

- We believe that for the optimal harmonic windows scheduling we can get a better lower bound than that given in Theorem 11. This is because we know the exact behavior of the algorithm described in the proof of Lemma 7 in the first stage of the recursion for any choice of x. In the work in progress, we plan to implement the asymptotic algorithm and would probably get better bounds than those implied by the combination technique for $h \leq 1789$.

- Is there another new technique that can yield a provably better asymptotic algorithm for the harmonic windows scheduling problem?

The optimal windows scheduling problem:

- Our asymptotic approximation uses $h(W) + e\ln(h(W)) + O(1)$ channels. Is there a $h(W) + O(1)$ solution or can that be shown to be impossible?
- Unlike the special case of the optimal harmonic scheduling problem, it is not clear how to define the dual problem of the general windows scheduling problem. One needs a notion of "throughput" to evaluate the performance of a solution for a given number of channels that cannot accommodate all the requested windows.

The media-on-demand and the push systems applications:

- Apparently, the windows scheduling problem does not require a fixed window size or that a page will appear only on one channel. Combinatorially, adding these two restrictions separately or combined could be very interesting. Moreover, one could imagine a media-on-demand model in which these restrictions could be useful. In this paper, we have two non-perfect schedules for the small values of $h = 4, 5$. Still, each page is scheduled only on one channel. On one hand, we conjecture that without these restrictions better solutions can be found. On the other hand, we conjecture that asymptotically these restrictions do not reduce the value of $N(h)$.
- Consider the case in which the receiving bandwidth of the clients is less than h. Capitalizing on the work of [16] and with the help of this paper results we can improve the tradeoff results between $N(r,h)$ and r/h where $N(r,h)$ is the maximum number of pages that can be scheduled on h channels provided that the receiving bandwidth of clients is $r < h$. This is a work in progress. The main idea is that having good solutions for $N(r)$ can improve the results for $N(r,h)$.
- In the push systems application, if the servers have a fixed number of channels, there is a need to device a strategy for "selling" windows. (This is related to the last item of the previous paragraph.)

Acknowledgments

Thanks to Dan Hoke for implementing the greedy algorithm and helping to calculate some of the values in the Table 1 and the result of the greedy algorithm for $h \leq 20$.

References

[1] S. Acharya, M. J. Franklin, and S. Zdonik. Dissemination-based Data Delivery Using Broadcast Disks. *IEEE Personal Communications*, Vol. 2(6), 50-60, 1995.

[2] M. H. Ammar and J. W.Wong. The Design of Teletext broadcast cycles. *Performance Evaluation*, Vol. 5(4), 235–242, 1985.

[3] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. In *Proceedings of the 9-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '98)*, 11–20, 1998.

[4] A. Bar-Noy, J. Naor, and B. Schieber. Pushing Dependent Data in Clients-Providers-Servers Systems. In *Proceedings of the 6-th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, 222–230, 2000.

[5] A. Bar-Noy, A. Nisgav, and B. Patt-Shamir. Nearly Optimal Perfectly-Periodic Schedules. In *Proceedings of the 20-th ACM Symposium on Principles of Distributed Computing (PODC '01)*, 107–116, 2001.

[6] L. Engebretsen and M. Sudan. Harmonic Broadcasting Is Optimal. In *Proceedings of the 13-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, 2002.

[7] V. Gondhalekar, R. Jain, and J. Werth. Scheduling on airdisks: Efficient access to personalized information services via periodic wireless data broadcast. The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-96-25. October 1996.

[8] K. A. Hua, Y. Cai, and S. Sheu Exploiting client bandwidth for more efficient video broadcast. In *Proceedings of the 7-th International Conference on Computer Communication and Networks (ICCCN '98)*, 848-856, 1998.

[9] K. A. Hua and S. Sheu. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 89-100, 1997.

[10] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, Vol. 44, No. 1, 100-105, 1998.

[11] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, Vol. 44, No. 1, 100-105, 1998.

[12] J. Pâris. A simple low-bandwidth broadcasting protocol for video-on-demand. In *Proceedings of the 8-th International Conference on Computer Communications and Networks (IC3N '99)*, 118-123, 1999.

[13] J. Pâris, S. W. Carter, and D. D. E. Long. Efficient broadcasting protocols for video on demand. In *Proceedings of the 6-th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, 127-132, 1998.

[14] J. Pâris, S. W. Carter, and D. D. E. Long. A low bandwidth broadcasting protocol for video on demand. In *Proceedings of the 7-th International Conference on Computer Communications and Networks (IC3N '98)*, 690-697, 1998.

[15] J. Pâris, S. W. Carter, and D. D. E. Long. A hybrid broadcasting protocol for video on demand. In *Proceedings of the IS&T/SPIE Conference on Multimedia Computing and Networking (MMCN '99)*, 317-326, 1999.

[16] J. Pâris and D. D. E. Long. Limiting the receiving bandwidth of broadcasting protocols for video-on-demand. In *Proceedings of the Euromedia Conference*, 107-111, 2000.

[17] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *ACM Multimedia Systems Journal*, Vol. 4(3), 197-208, 1996.

[18] S. Wolfram. Mathematica a System for Doing Mathematics by Computers. *Addison-Wesley Publishing Company.*