An Optimization Problem Related to VoD Broadcasting

Tsunehiko Kameda[1], Yi Sun[1], and Luis Goddyn[2]

[1] School of Computing Science
[2] Department of Mathematics,
Simon Fraser University, Burnaby, B.C. Canada V5A 1S6

Abstract. Consider a tree T of depth 2 whose root has s child nodes and the k^{th} child node from left has n_k child leaves. Considered as a round-robin tree, T represents a schedule in which each page assigned to a leaf under node k ($1 \leq k \leq s$) appears with period sn_k. By varying s, we want to maximize the total number $n = \sum_{k=1}^{s} n_k$ of pages assigned to the leaves with the following constraints: for $1 \leq k \leq s$, $n_k = \lfloor (m + \sum_{j=1}^{k-1} n_j)/s \rfloor$, where m is a given integer parameter. This problem arises in the optimization of a video-on-demand scheme, called *Fixed-Delay Pagoda Broadcasting*.

Due to the floor functions involved, the only known algorithm for finding the optimal s is essentially exhaustive, testing $m/2$ different potential optimal values of size $O(m)$ for s. Since computing n for a given value of s incurs time $O(s)$, the time complexity of finding the optimal s is $O(m^2)$. This paper analyzes this combinatorial optimization problem in detail and limits the search space for the optimal s down to $\kappa\sqrt{m}$ different values of size $O(\sqrt{m})$ each, where $\kappa \approx 0.9$, thus improving the time complexity down to $O(m)$.

1 Introduction

Recently, Bar-Noy et al. have formulated a combinatorial problem called the *windows scheduling problem* [1]. This problem is defined by positive integers c and $w_1, w_2, \ldots, w_n$, where c is the number of slotted channels and, for $i = 1, 2, \ldots, n$, a *window* of size w_i is associated with page i. A valid *schedule* assigns page i to slots such that it appears at least once in every window of w_i slots (not necessarily in the same channel).

Recently, there has been much interest in the broadcast-based delivery of popular videos, in order to address the scalability issue in video-on-demand (VoD). A VoD broadcasting scheme, called *Fixed-Delay Pagoda Broadcasting* (FDPB), was proposed by Pâris [5].[1]

A *channel* consists of a sequence of time slots of duration d (sec) each. The viewer initially downloads pages for md (sec) before s/he starts viewing the

[1] It has been implemented in a prototype video-on-demand (VOD) system [6]. Hollermann and Holzscherer [3] had also conceived a scheme similar to FDPB. A recent survey on VoD can be found in [4].

X. Deng and D. Du (Eds.): ISAAC 2005, LNCS 3827, pp. 116–125, 2005.

video. FDPB has the following constraints: (a) there is just one channel[2] that is divided into s subchannels, each subchannel consisting of every s^{th} slot of the channel; (b) $w_i = m+i-1$; (c) each page must be transmitted in one subchannel with a fixed period, and (d) all pages appearing in a given subchannel must be consecutive and have the same period. Our objective is to maximize the number of pages n that can be scheduled, by choosing the optimal value for s, which depends on m.

The author of [5] originally conjectured that $s = \sqrt{m}$ would be the optimal s value, but later found that it was not always the case among the examples he tested. Recently, Bar-Noy et al. [2] considered this problem and proposed a method whereby they could find the optimal value by testing $m/2$ different values of s. We analyze this dependency in detail in this paper and show that the optimal value of s is guaranteed to be one of roughly $0.9\sqrt{m}$ possible values. We thus can avoid time-consuming exhaustive search for the optimal s. The limited range for s reduced the running time of a computer program to compute the optimal value for s rather dramatically from a few hours down to a few seconds when $m = 10000$.[3]

The rest of the paper is organized as follows. Sect. 2 describes a useful tool, called the *round-robin tree*, introduced in [2]. In Sect. 3, we derive a formula for the optimal number s_{opt} of subchannels that maximizes the number of pages that can be scheduled under the constraints adopted by FDPB. We then find in Sect. 4 a small range for s in terms of m that needs to be searched, in order to find s_{opt}. Finally, in Sect. 5, we mention implications of our results, and also propose a new method of subchanneling such that a subchannel is divided into subsubchannels.

2 Preliminaries

2.1 Round-Robin Tree

The *round-robin (RR) tree* is a useful tool to represent a cyclic schedule [2]. Fig. 1 shows an example of a 2-level round-robin tree whose leaves are labeled by pages. A round-robin tree represents a schedule as follows:

1. Initially the root gets a "turn".
2. When a non-leaf node gets a turn, it passes the turn to its "next" child node. The leftmost child node gets a turn first and the order "next" means the next sibling to the right, wrapping around back to the leftmost child node.
3. When a leaf gets a turn, its associated page is scheduled and the turn goes back to the root. □

Example 1. Applying the above rules, it is seen that Fig. 1 represents the schedule, $\langle P_1, P_4, P_8, P_2, P_5, P_9, \ldots\rangle$. Note that pages P_1, P_2 and P_3 have period 9

[2] For the purpose of our discussion, this can be assumed without loss of generality. The general case is considered in the discussions section of this paper.

[3] Measured for a Java program running on a Pentium II CPU.

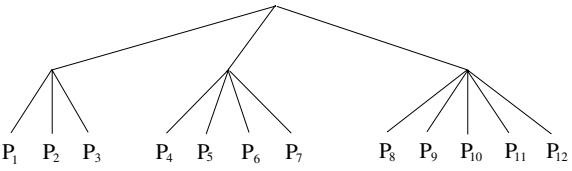

Fig. 1. A round-robin tree representation of FDPB with $s = 3$ subchannels

$(= 3s)$, pages $P_4, \ldots, P_7$ have period 12 $(= 4s)$, and pages $P_8, \ldots, P_{12}$ have period 15 $(= 5s)$. □

Lemma 1. [1] *Suppose the root of a 2-level RR tree T has s subtrees and for $k = 1, 2, \ldots, s$, the k^{th} subtree from left has n_k child leaves. Then T represents a schedule in which each page assigned to a leaf in the k^{th} subtree appears with period $n_k \times s$, and the minimum cycle of the schedule is given by $s \times \mathrm{LCM}(n_1, n_2, \ldots, n_k)$, where* LCM *stands for the least common multiple of the arguments.* □

2.2 Model

The k^{th} subtree of an RR tree T gets one "turn" out of every s "turns". We thus consider that the given channel is divided into s *subchannels* such that subchannel k consists of the time slots t satisfying $(t \bmod s) + 1 = k$, where time slots are numbered $t = 0, 1, 2, \ldots$ [5].

Lemma 2. [1, 5] *In FDPB with parameter m, in order for the viewer to be able to view the video continuously, page i must be broadcast (scheduled) at least once in each window of size $w_i = m + i - 1$.* □

Example 2. Let us suppose $m = 9$ and choose $s = 3$. Page P_1 can be scheduled in every $w_1 = 9 + 1 - 1 = 9^{th}$ slot. Since $s = 3$, subchannel 1 consists of every 3rd slot, and P_1 needs only 1/3 of it, and P_2 and P_3 can also be scheduled in subchannel 1, Thus we create a subtree with three leaves and label them by P_1, P_2 and P_3. See Fig. 1. These three pages will each have period 9 $(= 3s)$, which is adequate, since $w_2 > 9$ and $w_3 > 9$. Page P_4 must have period at most $w_4 = m + 4 - 1 = 12$ $(= 4s)$. Thus, we create the second subtree (representing subchannel 2) as shown in Fig. 1. Similarly for the last subtree. In summary, by dividing a channel into three subchannels, we can now pack 12, instead of 9 (when $s = 1$), pages in a channel. □

Let n_k denote the number of leaves of the k^{th} subtree. In order for P_1 to appear within every window of size $w_1 = m$, we must satisfy $sn_1 \leq m$ by Lemma 1. We thus get $n_1 = \lfloor m/s \rfloor$ as the maximum integer satisfying this inequality. Now that the first n_1 pages have been scheduled in subchannel 1, the next page, i.e., the $n_1 + 1^{st}$ page must have period at most $w_{n_1+1} = m + (n_1 + 1) - 1$, hence $sn_2 \leq m + n_1$, from which we get $n_2 = \lfloor (m + n_1)/s \rfloor$. In general, we have the following formula for n_k:

$$n_k = \lfloor (m + n_1 + n_2 + \cdots + n_{k-1})/s \rfloor \tag{1}$$

Let $n(m, s)$ denote the total number of pages that can be scheduled by a 2-level RR tree with s subtrees, i.e.,

$$n(m, s) = \sum_{k=1}^{s} n_k. \tag{2}$$

3 Optimization for FDPB

3.1 Problem

Fig. 2(a) plots $n(100, s)$ computed from (1) and (2) by varying s in the range $1 \leq s \leq 100$ (the rugged curve). It is seen that $s = 10$ maximizes $n(100, s)$.

In Fig. 2(b), $n(m, s)$ is plotted for many different values of m ($1 \leq m \leq 130$). For each value of m, a curve is drawn by varying s within the range $1 \leq s \leq m$. This can be considered as exhaustive search by which to find the optimal s that maximizes $n(m, s)$.[4] Note that the optimal value of s that maximizes $n(m, s)$ grows with m. Our main interest in this paper is to analyze the dependency of the optimal value of s on m in the hope of finding the optimal value without resorting to exhaustive search.

From what we have seen, we can use subtrees and subchannels almost synonymously. From now on, we will mainly use the term subtree. In reference to (1), for $k = 1, \cdots, s$, we define r_k ($0 \leq r_k \leq s - 1$) by

$$m + \sum_{j=1}^{k-1} n_j = s n_k + r_k. \tag{3}$$

We thus have for $k \geq 2$

$$\begin{aligned} n_k &= \lfloor (n_{k-1} s + r_{k-1} + n_{k-1})/s \rfloor = n_{k-1} + \lfloor (n_{k-1} + r_{k-1})/s \rfloor \\ r_k &= n_{k-1} + r_{k-1} \pmod{s}. \end{aligned} \tag{4}$$

In order to find the optimal value of s that maximizes $n(m, s)$ by differentiation, we try to approximate $n(m, s)$ by a function that doesn't contain any floor function. Let $\bar{r}$ denote the average of $\{r_k \mid k = 1, 2, \ldots, s\}$ in (3).

Lemma 3. *The total number of pages that can be assigned to the s subtrees is approximated by the following formula when s ($< m$) is sufficiently large:*

$$n(m, s) \approx (m - \bar{r}) \left((1 + \frac{1}{s})^s - 1 \right). \tag{5}$$

[4] The four dotted curves in Fig. 2(b) correspond to $m = 9, 21, 51$ and 128. (See Sect. 5.)

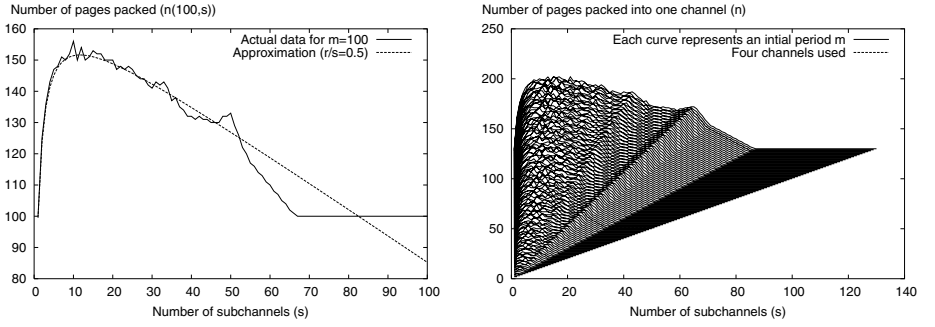

Fig. 2. (a) The rugged curve shows $n(100, s)$ for $1 \leq s \leq m = 100$. The smooth dashed curve is an approximation using (5) with $\bar{r}/s = 0.5$; (b) $n(m, s)$ for $m = 1, 2, \ldots, 130$.

Proof. Let $n_0 = m$ and rewrite (3) as follows: $r_k = \sum_{j=0}^{k-1} n_j - sn_k$, and hence $\frac{r_k}{s}(1+1/s)^{s-k} = (1+1/s)^{s-k} \sum_{j=0}^{k-1} n_j/s - (1+1/s)^{s-k} n_k$. Summing this from $k = 1$ to s, we get

$$\sum_{k=1}^{s} \frac{r_k}{s}(1+1/s)^{s-k} = \sum_{k=1}^{s}(1+1/s)^{s-k} \sum_{j=0}^{k-1} n_j/s - \sum_{k=1}^{s}(1+1/s)^{s-k} n_k.$$

We can rewrite the right hand side as follows: $\sum_{j=0}^{s-1}(n_j/s) \sum_{k=j+1}^{s}(1+1/s)^{s-k} - \sum_{k=1}^{s}(1+1/s)^{s-k} n_k = \sum_{j=0}^{s-1} n_j[(1+1/s)^{s-j} - 1] - \sum_{k=1}^{s}(1+1/s)^{s-k} n_k = m[(1+1/s)^s - 1] - n(m, s)$. We thus obtain

$$n(m, s) = m[(1+1/s)^s - 1] - \sum_{k=1}^{s} \frac{r_k}{s}(1+1/s)^{s-k}. \tag{6}$$

We notice that $n(m, 1) = n(m, m) = m$ and that $n(m, s)$ increases with s for small (relative to m) values of s. We now estimate $n(m, s)$ for large s ($< m$). For sufficiently large s, we assume that the remainders $r_1, \cdots, r_s$ are uniformly distributed in the range $[0,\ s-1]$. Substituting $r_i = \bar{r}$ into (6), we obtain (5). □

3.2 Solution

Corollary 1. *For sufficiently large s, the total number of pages $n(m, s)$ can be bounded as follows:*

$$(m - s + 1)\left((1 + \frac{1}{s})^s - 1\right) \leq n(m, s) \leq m\left((1 + \frac{1}{s})^s - 1\right)$$

Proof. Follows directly from Lemma 3 by setting $\bar{r} = s - 1$ (for the lower bound) and $\bar{r} = 0$ (for the upper bound). □

In order to find the optimal value of s that maximizes $n(m, s)$, one needs to evaluate (2) for s ranging $2 \leq s \leq m/2$ [2]. This is time-consuming when m gets large. In what follows, we show that the optimal solution s_{opt} can be found within a range containing $O(\sqrt{m})$ possible values of s. The following theorem shows how s_{opt} grows with m for large m.

Theorem 1. *The optimal number of subtrees s_{opt} grows roughly linearly with $\sqrt{m}$.*

Proof. Let $\bar{r} = a(s-1)$ in (5), where the range of parameter a is $0 < a < 1$. The optimal s satisfies the differential equation, $\frac{\partial n(m,s)}{\partial s} = 0$. Therefore, we have $(1+1/s)^s \left[ln(1+1/s) + \frac{s^2(1/s-(s+1)/s^2)}{s+1} \right] (m - a(s-1)) - a\left((1+1/s)^s - 1\right) = 0$. If s is sufficiently large, we can approximate the above equation as follows:

$$\frac{em/2 + \frac{23}{24}ea}{s^2} - (e-1)a \approx 0.$$

By solving the above quadratic equation, using the assumption $s^2 \gg a$, we obtain

$$s_{\text{opt}} \approx \sqrt{\frac{em}{2a(e-1)}}. \tag{7}$$

□

Fig. 3 plots the optimal number of subtrees, s_{opt}, computed by exhaustive search, varying s from 1 to m for each period m of the first page in the range up to 10000. It is observed that, despite the assumption $s \gg 1$ made to derive (5), practically all the data points are within an area bounded by $\sqrt{m} - 3$ and $1.54\sqrt{m}+6$. By reducing the range of search from $s \in [1, m/2]$ to $s \in [\max\{\sqrt{m} - 3, 1\}, 1.54\sqrt{m} + 6]$, the execution time of our search program for all values of m between 1 and 50000 went down rather drastically from more than one day to less than a minute. This range is roughly $(1.54 - 1)\sqrt{m} \approx 0.54\sqrt{m}$.

4 Theoretical Bounds

Although the results obtained in the previous section are quite satisfactory, there is no guarantee that nothing strange will happen beyond $m = 10000$. In order to dispel such misgivings, we shall derive theoretical upper and lower bounds in this section. They are shown in Fig. 3 as the top and bottom curves.

The formula (7) is not directly useful, since $a = a(m, s)$ is a function of s. In order to overcome this problem, let S denote a range for s such that $s_{\text{opt}} \in S$ for large m. Define $a^{up} = Sup_{s \in S}\{a(m, s)\}$ and $a^{low} = Inf_{s \in S}\{a(m, s)\}$ Then from (7), we have

$$\sqrt{\frac{em}{2a^{up}(e-1)}} \leq s_{\text{opt}} \leq \sqrt{\frac{em}{2a^{low}(e-1)}}. \tag{8}$$

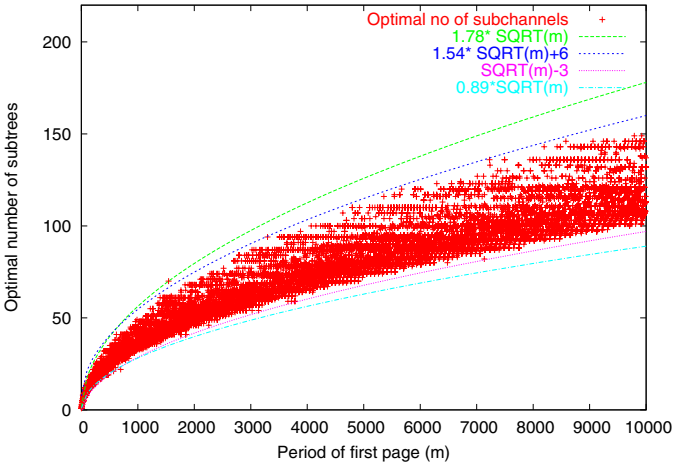

Fig. 3. The optimal number of subtrees ($s_{\rm opt}$) vs. the period (m) of the first page. The actual optima are plotted as data points.

Since $a < 1$, we obviously have $a^{up} < 1$. In the rest of this section, we shall find a lower bound on a^{low}. Fig. 4(a) plots the computation results for $a(10000,\ s)$ for s in the range $1 \le s \le m = 10000$. The initial part of the graph is blown up in Fig. 4(b).

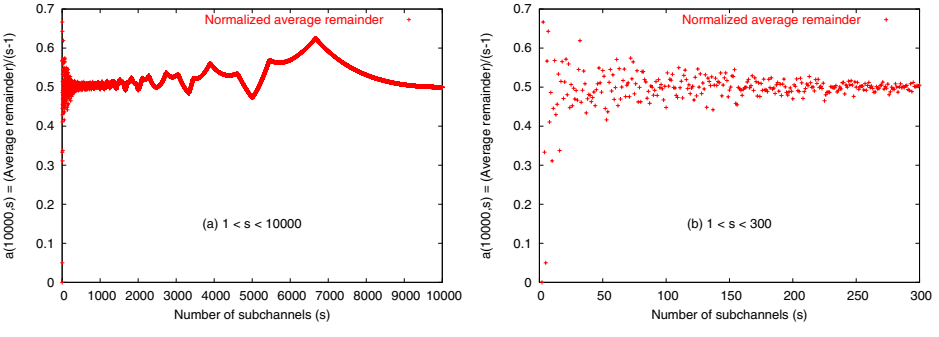

Fig. 4. (a) Quantity $a(10000, s) = \bar{r}/(s-1)$ vs. s $(1 \le s \le 10000)$; (b) Quantity $a(10000, s) = \bar{r}/(s-1)$ vs. s $(1 \le s \le 300)$

In order to find a^{low} for given m, we want to investigate how the remainder changes in the vicinity of $s = \sqrt{m}$, where the optimal value for s is to be found.

Example 3. Fig. 5 illustrates the case where $m = 10000$ and the number of subtrees $s = 150$. (The optimal number of subtrees in this case is $s_{\rm opt} = 132$.) The horizontal axis represents the 150 subtrees, i.e., subtrees $k = 1$ to $k =$

150. Each number in Fig. 5 is n_k of (1) for some k, and its height represents r_k of (3). The lower horizontal line is the average of all the remainders $\{r_k \mid k = 1, 2, \ldots, s\}$, while the upper horizontal line is at height $(s-1)/2$. The dotted curves (each "dot" is represented by symbol "+") in Fig. 5 represent the difference $|r_{k+1} - r_k|$ for $k = 1, 2, \ldots, s-1$. For example, let index k be such that $n_k = 137$ and $r_k = 5$. (Of the two points in Fig. 5 that are labeled by 137, this point is the one close to the bottom.) Then by (4) we can compute $n_{k+1} = 137 + \lfloor (137+5)/150 \rfloor = 137$ and $r_{k+1} = 137+5 \pmod{150} = 142$, which correspond to the other point labeled by 137. The difference $r_{k+1} - r_k = 137$ is seen as a "+" just below this 137. Similarly, we can compute $n_{k+2} = 138$ and $r_{k+2} = 129$, and hence $|r_{k+2} - r_{k+1}| = 8$. This explains the position of the point labeled by 138 and the height of the next "+". □

One striking feature in Fig. 5 is the presence of quadratic curves (parabolas).We now examine the cause of the parabolas observed in the above example. For $m = 10000$ and $s > \sqrt{m} = 100$, after computing $n_1, n_2, \ldots$, suppose we reach $i-2$ such that $n_{i-2} = s-1$ and $r_{i-2} = r$ for some r. Using (4), we get $n_{i-1} = s - 1 + \lfloor (s-1+r)/s \rfloor = s, r_{i-1} = r-1$; $n_i = s+1, r_i = r-1$; ($n_i = s+1$ appears at the bottom of the right parabola in Fig. 5. If s is chosen too large, this parabola will move to the right out of range.) $n_{i+1} = s+2, r_{i+1} = r = (r-1)+1$; $n_{i+2} = s+3, r_{i+2} = (r-1)+1+2$, and so forth, and in general,

$$r_{i+j} = (r-1) + \sum_{h=1}^{j} h = (r-1) + j(j+1)/2, \tag{9}$$

for $0 \le j \le p$, where p is the maximum number such that $(r-1) + p(p+1)/2 \le s-1$. It is seen that r_{i+j} is a quadratic function of j.

Lemma 4. *For sufficiently large m, if $s > \sqrt{m}$ then the average remainder $\bar{r} > (s-1)/4$.*

Proof. Fix the value $s > \sqrt{m}$ near $\sqrt{m}$,[5] and assume the worst case, where the remainders are concentrated near 0, i.e., the remainder in (9) has the form $r_{i+j} = j(j+1)/2$ (i.e., $r = 1$) for $0 \le j \le p$. Thus the remainders take one of the $p+1$ values, $0, 1, 3, \ldots, p(p+1)/2$, because this leads to the most skewed distribution of remainders towards 0. Note that p satisfies $(p+1)(p+2)/2 > s-1$. Then the average value of all the $p+1$ remainders is given by $R = (1/(p+1)) \sum_{j=1}^{p} j(j+1)/2 = p(p+2)/6$. Using $(p+1)(p+2)/2 > s-1$, we can derive $R > (1 - 1/(p+1))(s-1)/3 > [1 - 1/(\sqrt{2(s-1)} - 1)](s-1)/3$. For $s \ge 14$, we have $[1 - 1/(\sqrt{2(s-1)} - 1)] > 3/4$, and therefore $\bar{r} > R > (s-1)/4$. □

Quadratic growth is not clearly discernible in the middle part of Fig. 5, but it is there. For example, there is a parabolic curve segment on which the points

[5] As commented earlier, if s is chosen too large, the parabola that contains a point labeled by $n_k = s$ at its bottom won't appear. In such a case, see the comments after this lemma.

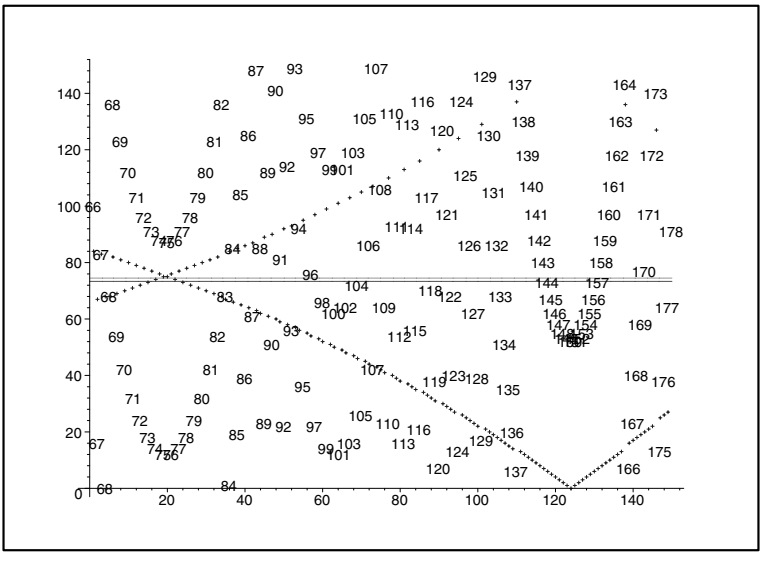

Fig. 5. Above k ($1 \leq k \leq s = 150$) on the horizontal axis, a point with label n_k is plotted at height r_k

labeled by 95, 96 and 97 lie. It is easy to see that in general the average of the remainders on such a parabolic curve segment is larger than $(s-1)/4$. In particular, if there are only two points on a parabolic curve segment, then their difference in height must be $> (s-1)/2$, and hence their average should be $> (s-1)/4$. There may be one or two points, e.g., the point labeled by 101 near the bottom of the graph, which do not share a parabolic curve segment with any other points. But their influence on lowering the average remainder can be compensated for by other large remainders. Note that on the left part of Fig. 5 there are parabola-like patterns, but they are not parabolas in the sense the term is used here. Two points labeled by 80, for example, form a parabolic curve segment.

Theorem 2. *For sufficiently large m, the optimal number of subtrees $s_{\rm opt}$ that maximizes $n(m,s)$ can be bounded as follows:*

$$\sqrt{\frac{em}{2(e-1)}} < s_{\rm opt} < \sqrt{\frac{2em}{(e-1)}}.$$

Proof. Lemma 4 implies $a^{low} > 1/4$. Plug it and $a^{up} < 1$ into (8). □

According to the above theorem, only $\sqrt{2em/(e-1)} - \sqrt{em/2(e-1)} \approx 1.78\sqrt{m} - 0.89\sqrt{m} = 0.89\sqrt{m}$ different values of s need be tested to find $s_{\rm opt}$. The bounds given by the above theorem are shown as the top and bottom curves in Fig. 3.

5 Discussions

Let us consider the general case, where we have c channels, $C_1, C_2, \ldots, C_c$. Suppose the maximum period of the initial page for channel C_1 is m_1. Then we have $m_2 = m_1 + (i_1 + 1) - 1$ for first page of channel C_2, if pages 1 to i_1 are packed into channel C_1. Thus by varying the parameter m in our previous analysis, we can determine how many pages can be packed into the second, third, ... channels. For example, the four dotted curves in Fig. 2(b) correspond to $m = 9, 21, 51$ and 128. The curve for $m_1 = 9$ reaches its peak 12 at $s = 3$, which implies that 12 pages can be packed into C_1 if the period of the first page is 9 and three subtrees are used. Thus page 13 is the first page to be packed in C_2, and therefore we should look at the curve for $m_2 = 9 + 13 - 1 = 21$. This curve has the peak value of 30, and thus 30 pages can be packed into C_2, and so forth.

We could use recursive subchanneling. Namely, we first divide a channel into s subchannels of equal bandwidth as before, and then further divide each of the s subchannels into subsubchannels, and so forth. In other words, instead of a 2-level RR tree, we use a RR tree with 3 or higher levels. It turns out that we are able to fit only slightly more pages into a channel for some values of m. Recursive subchanneling becomes more beneficial for larger values of m.

Although the problem addressed in this paper is rather special, we believe the approach we used could be applied to many other optimization problems that involve the floor or ceiling function.

Acknowledgement

We thank the members of the Distributed Computing Laboratory in the School of Computing Science at Simon Fraser University for stimulating discussions.

References

1. Bar-Noy, A., Ladner, R.E.: Windows scheduling problems for broadcast systems. Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (2002) 433–442
2. Bar-Noy, A., R.E. Ladner, R.E., Tamir, T.: Scheduling Techniques for Media-on-Demand. Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (2003) 791–800
3. Hollmann, H.D.L., Holzscherer, C.D.: Philips Tech. Rept. (1991) European Patent (1991) and US patent No. **5524271** (1995)
4. Kameda, T., Sun, T.: Survey on VoD broadcasting schemes. School of Computing Science, SFU (2003) `http://www.cs.sfu.ca/~tiko/ publications/ VODsurveyPt1.pdf`
5. Pâris, J.-F.: A fixed-delay broadcasting protocol for video-on-demand. 10th Int'l Conf. on Computer Communications and Networks (2001) 418–423
6. Thirumalai, K., Pâris, J.-F., Long, D.D.E.: Tabbycat: an inexpensive scalable server for video-on-demand. Proc. IEEE International Conference on Communications (2003) 896–900