

Nearly Optimal Perfectly Periodic Schedules

Amotz Bar-Noy¹, Aviv Nisgav², Boaz Patt-Shamir²

¹ Dept. of Computer and Information Science, CUNY, Brooklyn, NY 11210, USA. e-mail: amotz@sci.brooklyn.cuny.edu.

² Dept. of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel. e-mail: {aviv,boaz}@eng.tau.ac.il.

The date of receipt and acceptance will be inserted by the editor

Summary. We consider the problem of scheduling a set of pages on a single broadcast channel using time-multiplexing. In a *perfectly periodic schedule*, time is divided into equal size *slots*, and each page is transmitted in a time slot precisely every fixed interval of time (the *period* of the page). We study the case in which each page i has a given demand probability w_i , and the goal is to design a perfectly periodic schedule that minimizes the average time a random client waits until its page is transmitted. We seek approximate polynomial solutions. Approximation bounds are obtained by comparing the costs of a solution provided by an algorithm and a solution to a relaxed (non-integral) version of the problem.

A key quantity in our methodology is a fraction we denote by a_1 , that depends on the maximum demand probability: $a_1 \stackrel{\text{def}}{=} \sqrt{\max_i \{w_i\}} / \sum \sqrt{w_i}$. The best known polynomial algorithm to date guarantees an approximation of $\frac{3}{2} + \frac{3}{2}a_1$. In this paper, we develop a tree-based methodology for perfectly periodic scheduling, and using new techniques, we derive algorithms with better bounds. For small a_1 values, our best algorithm guarantees approximation of $1 + \frac{\sqrt[3]{3a_1}}{1 - \sqrt[3]{3a_1}}$. On the other hand, we show that the integrality gap between the cost of any perfectly periodic schedule and the cost of the fractional problem is at least $1 + a_1^2$. We also provide algorithms with good performance guarantees for large values of a_1 .

Key words: Perfectly periodic scheduling – Broadcast disk – Asymmetric communication – Maintenance scheduling

1 Introduction

The prevalence of portable information devices today—as well as their omni-presence tomorrow—is unquestionable. One of the major problems of this technology is power supply, since these gadgets may weigh only few grams, leaving very little room for batteries. In this paper, we investigate one aspect of this area: we present

algorithms to compute schedules that can significantly reduce the power consumption of a mobile client while waiting for a certain piece of information (*page*) to be broadcast by a stationary server. We give scheduling algorithms that improve the best known results, and develop algorithmic tools for tree embedding that have other important applications such as fair time sharing.

Our basic premise is that the server must broadcast pages in a very organized fashion. Informally, a schedule is called *perfectly periodic* if each page has a single period τ such that this page is transmitted exactly every τ time units. Perfect periodicity allows clients to save significantly on energy, since it reduces “busy waiting” (see below). From the theoretic viewpoint, perfect periodicity leads to interesting algorithmic questions. Our starting point is the following scheduling problem, stated in the language of *broadcast disks* [2]. The input to the problem is a set $\{1, \dots, M\}$ of pages, with a given demand probability w_i for each page i . The goal is to find a perfectly periodic schedule that minimizes the average waiting time of a random client. Suppose that page i is scheduled every τ_i time units in a given schedule. Then the average waiting time for a client for page i is $\frac{\tau_i}{2}$; the probability for a random client to wait for page i is w_i , and hence, the average waiting time for a random client is $\frac{1}{2} \sum_{i=1}^M w_i \tau_i$. As we shall see later, one can easily derive from the given demand probabilities a set of “ideal” requested periods that will minimize the average waiting time; however, these desired periods will not be feasible by a perfectly periodic schedule in general (for example, they may be fractional). The problem addressed in this paper, therefore, is the following: given a set of demand probabilities (or, equivalently, requested periods), construct a perfectly periodic schedule whose periods approximate the ideal periods as closely as possible.

1.1 Motivation

There are many reasons that support the thesis that perfectly periodic schedules are useful. We provide two examples here.

Broadcast disks [2]: In this setting, a powerful server (e.g., a satellite) broadcasts data pages to mobile clients awaiting their desired pages. In an arbitrary broadcasting schedule, a client may have to “busy-wait” for its page, i.e., actively listen to the server until the desired page is broadcast. This difficulty can be alleviated if the schedule is perfectly periodic, since the client can compute when its desired page is broadcast next. This can be done in at least two ways. First, it suffices for the client to know the time of any two consecutive broadcasts of its page. The second way is to take advantage of the Khanna-Zhou indexing scheme [20]. In this scheme, the server also broadcasts *index* pages from time to time: the index contains information about when the data pages are broadcast. This allows the client to busy-wait only for the first index page. The point is that the only known efficient indexing schemes are applicable when the underlying schedule is perfectly periodic [20]. In any case, it follows that if the schedule is perfectly periodic, client devices can avoid busy waiting, allowing them to decrease their power consumption. For example, they can shut down their receivers and switch their CPU to “doze mode,” in which the power consumption is miniscule [11, 17]. Thus, perfectly periodic schedules may be sometimes preferable to other schedules even if they cause a higher average waiting time for clients since they allow most of the waiting time to come virtually for free, in terms of client resources.

Fair time sharing: Another motivation for perfect periodicity is the more general problem of *time sharing*. In this setting, an indivisible resource is to be shared by M clients by means of time multiplexing. Client i should get a *share* (or *frequency*) of $1/\tau_i$ of the resource such that $\sum_{i=1}^M 1/\tau_i \leq 1$. The difficulty is that the schedule should also be *fair*: intuitively, fairness means that the number of time slots client i waits should be as close as possible to τ_i . A canonical example is the classical *chair-person assignment problem* [25]. Several fairness criteria were considered in the past. For example, in the *prefix criterion*, the requirement is that in any prefix of T slots, each client i gets either $\lfloor T/\tau_i \rfloor$ or $\lceil T/\tau_i \rceil$ slots [21]. Since the number of slots is integral, this seems to be the best possible. Indeed, there exists a schedule that meets the prefix fairness requirement [25]. Still, the length of the interval between two appearances of the same client could be as large as $2\tau_i$, i.e., twice the average interval. The strictest fairness requirement is to find a perfectly periodic schedule with periods τ_i . However, such a schedule is not always possible, as implied by the Chinese Remainder Theorem. For example, there is no perfectly periodic schedule with $\tau_1 = 2$ and $\tau_2 = 3$. Worse, it is NP-complete to decide whether a given set of frequencies admits a perfectly periodic schedule when $\sum_{i=1}^M 1/\tau_i \leq 1$ [8]. One way to overcome this difficulty is to allow different gaps between consecutive appearances of a client in the schedule, while insisting on exact average frequency allocation. This is the approach taken by many researchers in the area of broadcast disks and related problems. In this paper, we explore the other alternative: we insist on maintaining the perfect periodicity

property while relaxing the exact average frequency allocation requirement. Namely, we consider perfectly periodic schedules in which the frequency assigned to a client may be different than its demand. The objective of a good schedule in this case is to minimize the weighted average of the ratios between the requested and the allocated frequencies. More precisely, suppose that each client i requires a period of τ_i such that $\sum_{i=1}^M 1/\tau_i = 1$. Let S be a perfectly periodic schedule in which the period of client i is τ'_i . The ratio τ'_i/τ_i measures how close the assigned period is to the requested period. Giving weight to clients proportionally to their requested share (i.e., giving client i weight $1/\tau_i$), leads to the following objective. Find a perfectly periodic schedule that minimizes

$$\sum_{i=1}^M \frac{1}{\tau_i} \cdot \frac{\tau'_i}{\tau_i} = \sum_{i=1}^M \frac{\tau'_i}{\tau_i^2}.$$

This turns out to be exactly the target function of perfectly periodic scheduling for minimizing the the average waiting time (see Lemma 2 below).

1.2 Related work

To the best of our knowledge, this is the first work on perfectly periodic schedules *per se*. Most prior research focused on minimizing the average waiting time even at the cost of non-perfectly periodic schedules, though some of the known results are based on perfectly periodic schedules. The main motivation for our work comes from wireless communication. In particular, the broadcast disks problem and its numerous variants [17, 1, 2, 26, 24, 8, 18, 7, 19], and the Teletext problem [3, 4]. Closely related problems from the operation research area include the machines maintenance problem [27, 5, 6], the pinwheel problem [16, 10], multi-item replenishment and other inventory problems [13, 23, 14]. Below, we briefly mention a few quantitative relevant results.

In [3, 4], Ammar and Wong consider the problem of minimizing the mean response time in Teletext systems, which is equivalent to the problem of minimizing the average waiting time in the broadcast disks problem. They present a 2-approximation algorithm. In [5], Anily *et al.* study the maintenance scheduling problem and the broadcast disks problem. They present a 2-approximation algorithm for the broadcast disks problem. This algorithm, with a slight change, is a 2-approximation perfectly periodic schedule.

Khanna and Zhou [20] define *waiting time* as the total time until the client gets its requested page, and *tuning time* as the total time the client is *active* while waiting (a.k.a. busy waiting time). To minimize tuning time, they design an indexing scheme, which works only for perfectly periodic schedules. They also give the best approximation of perfectly periodic scheduling to date. Their algorithm achieves mean waiting time of at most 1.5 times the optimal waiting time (plus an additive $O(\log M)$ term). In [17], Imielinski *et al.* empirically study the impact of organization and indexing of data in broadcast disks on power consumption.

For schedules that are not perfectly periodic, the following results are known. Liu and Layland [21] define periodic scheduling to be one where a job (page) with period τ is scheduled exactly once in each time interval of the form $[(k-1)\tau, k\tau-1]$ for any integer k . In the generalized model, each job has an integral length and there may be multiple servers; for this case, Baruah *et al.* [9] propose the notion of “Pfair schedules,” where the number of slots allocated to a job whose share is b in any prefix of t time slots is either $\lfloor b \cdot t \rfloor$ or $\lceil b \cdot t \rceil$.

In [8], using the Golden Ratio schedule [15], Bar-Noy *et al.* give a $\frac{9}{8}$ -approximation. In their algorithm, the gaps between two consecutive occurrences of a page may take one of three distinct values. They also prove that finding the optimal perfectly periodic schedule when $\sum_{i=1}^M 1/\tau_i \leq 1$ is NP-hard. Finally, recently, Kenyon *et al.* [19] present a PTAS for the broadcast disks problem. Their solution is not perfectly periodic in general.

1.3 Our Results

In this paper, we study perfectly periodic schedules for the broadcast disks problem. We introduce the methodology of tree-based schedules, which are essentially hierarchical composition of round-robin schedules. In this methodology, we break the schedule computation into two parts. One problem is how to embed a given set of frequencies in a tree: once it is embedded in a tree, deriving a schedule is easy. The other problem is how to manipulate a given set of frequencies into another, so that the resulting set can be embedded in a tree, while minimizing the target function (average waiting time). Our technical results therefore fall into two categories:

- First, we develop a few tree embedding techniques. These techniques are stated as sufficient conditions on a frequency set to be embeddable in a tree.
- Then we present algorithms to produce the perfectly periodic schedules by manipulating the given set of requested frequencies into another that satisfies one of the sufficiency conditions. The central goal in these algorithms is to keep the average period close to optimal.

We evaluate the performance of our algorithms by comparing them to the solution to non-linear program, which we call the *relaxed* problem. Informally, in the relaxed problem, the target function is the same, but the only constraint is that the sum of the frequencies is at most 1. We know how to calculate the *relaxed solution*, namely the optimal solution to the relaxed problem. Since the cost of the relaxed solution is a lower bound for the cost of any schedule, the worst-case ratio between the costs of the schedule produced by a given algorithm and the relaxed solution is an upper bound on the approximation factor of the algorithm. We note that unlike the broadcast disks problem, in the problem of fair time sharing, the desired frequencies are given, and therefore the approximation factor for this problem, by definition and Lemma 2 below, is *exactly* the worst-case ratio between

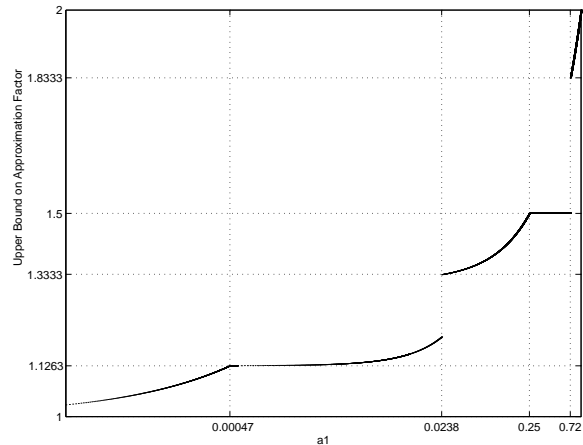


Fig. 1. The upper bound on the approximation factor as a function of a_1 . Note that the x scale is logarithmic. The segment of $a_1 < 0.00047$ is attained by Algorithm A; the segment of $a_1 \in [0.00047, 0.0238]$ is attained by Algorithm B; the segments of $a_1 \in [0.0238, 0.25]$ and $a_1 \in [0.7176, 1]$ are attained by Algorithm C; and the segment of $a_1 \in [0.25, 0.7176]$ is attained by Algorithm D.

the cost of the algorithm and the cost of the relaxed solution.

In this paper we show that the approximation factor depends on the quantity $0 < a_1 \leq 1$ which is the frequency allocated to the largest demand in the relaxed solution. Formally, a_1 is defined by

$$a_1 \stackrel{\text{def}}{=} \frac{\sqrt{\max_i \{w_i\}}}{\sum_{i=1}^M \sqrt{w_i}}.$$

Roughly speaking, our results show that the cost of a perfectly periodic schedule is one plus a function of a_1 times the optimal cost of the relaxed problem. In more detail, our four algorithms are as follows (ordered in increasing approximation factors for small value a_1).

- Algorithm A: For $a_1 < \frac{1}{3}$, the approximation factor is at most $1 + \frac{\sqrt[3]{3a_1}}{1 - \sqrt[3]{3a_1}}$.
- Algorithm B: For $a_1 < \frac{1}{42} \approx 0.024$, the approximation factor is at most $\frac{9}{8 - 20a_1}$.
- Algorithm C: For all values of a_1 , the approximation factor is at most $\frac{4}{3} + \frac{2}{3}a_1$.
- Algorithm D: For $a_1 < \frac{19 - \sqrt{37}}{18} \approx 0.72$, the approximation factor is at most $\frac{3}{2}$.

Each of these bounds is the best known for some value of a_1 (see summary in Figure 1).¹ All our algorithms are polynomial. Algorithm D is omitted from this paper, as it is rather technical and contains only few new ideas; the interested reader can find it in [22].

We complete the picture by showing that the dependency of the approximation factor on a_1 is unavoidable using the relaxed solution to bound the optimal

¹ A trivial modification of the algorithm in [20] gives an approximation factor of $\frac{3}{2} + \frac{3}{2}a_1$. However, this result is still inferior to Algorithm C for all values of $a_1 < 1$.

cost. Specifically, we prove that there exist sequences for which no perfectly periodic schedule can have cost less than $1 + a_1^2$ times the cost of the best relaxed solution. Moreover, for schedules that can be represented as trees (as are all of our upper bounds), we improve this lower bound to $1 + 0.8a_1$. These bounds can be viewed as bounds on the integrality gap of the problem.

1.4 Paper organization

The remainder of the paper is organized as follows. In Section 2, we formalize the problem and present a known lower bound on average waiting time. We also prove a new upper bound on the approximation factor of perfectly periodic schedules. In Section 3, we describe our tree representation for perfectly periodic schedules and present embedding algorithms for certain sets of input frequencies. In Sections 4, 5, and 6, we describe Algorithms C, B, and A respectively, ordered by improving approximation bounds (and increased complexity of description). In Section 7, we present our lower bounds. Concluding comments are given in Section 8.

2 Model and Preliminaries

In this section, we formally define the problem and present some preliminary results. The results in this section are known [3] with the notable exception of Lemma 3, which is the key to the analysis of our algorithms.

We are given a set of M pages $\mathcal{P} = \{1, \dots, M\}$. A schedule $S = \langle A_1, A_2, A_3, \dots \rangle$ for \mathcal{P} is an infinite sequence such that $A_t \in \mathcal{P} \cup \{\perp\}$. If $A_t = i$, then we say that page i is *scheduled* at time t by S , and if $A_t = \perp$, then no page is scheduled at time t by S . A schedule S is called *perfectly periodic* if for each page $i \in \mathcal{P}$ there exists a natural number τ_i , called the *period* of i , with the following property: for any $t \geq 1$, if page i is scheduled at time t , then page i is scheduled at time $t + \tau_i$, and page i is not scheduled at times $t+1, t+2, \dots, t + \tau_i - 1$. The reciprocal of the period, $f_i \stackrel{\text{def}}{=} \frac{1}{\tau_i}$, is called the *frequency* of i . We sometimes prefer to use f_i over τ_i . A schedule $S = \langle A_1, A_2, \dots \rangle$ is called *cyclic* if it is an infinite concatenation of a finite sequence $\langle A_1, \dots, A_T \rangle$. Such a sequence $\langle A_1, \dots, A_T \rangle$ is said to *generate* S . We have the following simple property.

Proposition 1. *If a schedule S for set of pages \mathcal{P} is perfectly periodic, then it is cyclic.*

Proposition 1 follows from the fact that S is generated by its first T elements, where T is the least common multiple of $\{\tau_i \mid i \in \mathcal{P}\}$.

We assume that clients arrive at uniform random times during the schedule. Given a schedule S , we say that the *waiting time* for page i for a client that arrives at time t is the number of time slots, starting from t , until i is scheduled in S . This number is not necessarily integer if the arrival time of the client is in the middle of a slot. Note that in a perfectly periodic schedule, if a page

has a period τ , then the expected waiting time for that page is $\tau/2$, assuming that all arrival times are equally likely. We remark that since we consider perfectly periodic schedules, the concept of a uniform random arrival time is well defined as a corollary of Proposition 1.

The *Periodic Approximate Scheduling* problem is defined as follows. We are given a set \mathcal{P} of pages with a *demand probability* $w_i > 0$ for each page $i \in \mathcal{P}$ such that $\sum_{i \in \mathcal{P}} w_i = 1$. Throughout this paper, we assume w.l.o.g. that $w_i \geq w_{i+1}$ for all i . The goal is to produce a perfectly periodic schedule S for \mathcal{P} that minimizes the average waiting time $W(S)$, defined by

$$W(S) = \sum_{i \in \mathcal{P}} w_i \frac{\tau_i}{2} = \frac{1}{2} \sum_{i \in \mathcal{P}} \frac{w_i}{f_i},$$

where τ_i denotes the period of page i in S .

Let S^* denote a perfectly periodic schedule that minimizes the average waiting times for a given set of pages \mathcal{P} and their demand probabilities $\{w_i \mid i \in \mathcal{P}\}$. Let the periods in S^* be $\tau_i^{S^*}$ for each page $i \in \mathcal{P}$. Given another perfectly periodic schedule S for \mathcal{P} , where τ_i^S is the period of i in S , we define the *approximation factor* of S , denoted $A(S)$, to be the ratio between the average waiting time of S and the average waiting time of S^* . Formally,

$$A(S) \stackrel{\text{def}}{=} \frac{W(S)}{W(S^*)} = \frac{\frac{1}{2} \sum_{i=1}^M w_i \tau_i^S}{\frac{1}{2} \sum_{i=1}^M w_i \tau_i^{S^*}} = \frac{\sum_{i=1}^M w_i \tau_i^S}{\sum_{i=1}^M w_i \tau_i^{S^*}}.$$

We do not know how to find the cost of the optimal schedule S^* , so we resort to bounding it from below. This is done by considering the following continuous relaxation of the problem [5,8]. Specifically, we remove most feasibility constraints and leave only the constraint that the sum of the reciprocals of the positive periods cannot exceed 1. In particular, the periods τ_i do not have to be integers. Formally, we have the following non-linear program. Given w_1, \dots, w_M , find τ_1, \dots, τ_M that

$$\begin{aligned} & \text{minimize } \frac{1}{2} \sum_{i=1}^M w_i \tau_i \\ & \text{subject to } \sum_{i=1}^M \frac{1}{\tau_i} \leq 1, \text{ and } \tau_i > 0 \text{ for all } i. \end{aligned} \quad (1)$$

Obviously, an optimal solution to (1) is a lower bound on the average waiting time of any perfectly periodic schedule. The following well-known result gives a closed-form solution to (1).

Lemma 1.

1. *The optimal solution of the non-linear program (1) is $\tau_i^{s^*} = \frac{\sum_{j=1}^M \sqrt{w_j}}{\sqrt{w_i}}$ for $i = 1, \dots, M$.*
2. *Let S^* be an optimal perfectly periodic schedule for a set of pages \mathcal{P} with demand probabilities $\{w_i\}$. Then $W(S^*) \geq \frac{1}{2} \left(\sum_{i \in \mathcal{P}} \sqrt{w_i} \right)^2$.*

The proof is based on Lagrangian relaxation (see, e.g., [5,8,22]).

In general, there may not exist any perfectly periodic schedule whose periods are given by the optimal solution to the relaxed problem (1). Our approximation algorithms apply different methods of rounding the periods obtained from the relaxed solution to get a feasible schedule. We use the lower bound of Lemma 1 to get an upper bound on the approximation factor of the resulting schedules. Moreover, this lemma gives rise to the definition of the quantities a_i as the optimal frequencies in the relaxed problem:

$$a_i \stackrel{\text{def}}{=} \frac{\sqrt{w_i}}{\sum_{j=1}^M \sqrt{w_j}}.$$

Let $C(S)$ be the cost of a schedule S defined as follows:

$$C(S) = \sum_{i=1}^M \tau_i a_i^2 = \sum_{i=1}^M \frac{a_i^2}{f_i}.$$

Using this notation, the following lemma provides a simple upper bound on the approximation factor of any given schedule. The proof is by some algebraic manipulations on the definition of $A(S)$ and Lemma 1.

Lemma 2. *For all perfectly periodic schedules S , $A(S) \leq C(S)$.*

Fix a schedule S . Let $\rho_i \stackrel{\text{def}}{=} \frac{a_i}{f_i}$, where f_i is the frequency of i in S . By definition, $C(S)$ is a weighted average of the ρ_i 's (the weight of ρ_i is a_i). It is therefore obvious that $C(S)$ – and hence $A(S)$ – are bounded by $\max\{\rho_i \mid i = 1, \dots, M\}$. The following algebraic lemma shows a better bound, in terms of the “leftover capacity” (the fraction of slots not allocated to any page). It serves as the key to our bounds on the approximation factor of the algorithms we develop. It is convenient for us to state it abstractly, without assuming the full structure of a schedule.

Lemma 3. *Let a_1, \dots, a_M and f_1, \dots, f_M be positive real numbers such that $\sum_{i=1}^M a_i = 1$ and $\sum_{i=1}^M f_i \leq 1$. Let $\Delta = 1 - \sum_{i=1}^M \frac{a_i}{f_i}$. If $\underline{\rho}$ and $\bar{\rho}$ are such that $\underline{\rho} \leq \frac{a_i}{f_i} \leq \bar{\rho}$ for all i , then*

$$\sum_{i=1}^M \frac{a_i^2}{f_i} \leq \underline{\rho} + \bar{\rho} - \underline{\rho}\bar{\rho} + \underline{\rho}\bar{\rho}\Delta.$$

Proof. If $\underline{\rho} = \bar{\rho} = \rho$ then $f_i = \frac{a_i}{\rho}$ for all i , and hence $\Delta = 1 - \frac{1}{\rho}$. Therefore,

$$\bar{\rho} + \underline{\rho} - \underline{\rho}\bar{\rho} + \underline{\rho}\bar{\rho}\Delta = 2\rho - \rho^2 + \rho^2 \left(1 - \frac{1}{\rho}\right) = \rho.$$

Obviously in this case $\sum_{i=1}^M \frac{a_i^2}{f_i} = \sum_{i=1}^M \rho a_i = \rho$, and the lemma holds.

Assume now that $\underline{\rho} < \bar{\rho}$. Fix the f_i values and consider the a_i -s as variables. First, we claim that for all i ,

$$\frac{a_i^2}{f_i} \leq \underline{\rho}^2 l_i + \bar{\rho}^2 h_i, \quad (2)$$

where l_i and h_i are the solution to the following inequalities:

$$\begin{aligned} l_i + h_i &= f_i \\ \underline{\rho} l_i + \bar{\rho} h_i &= \rho_i f_i. \end{aligned} \quad (3)$$

The intuition is that because we only have bounds on ρ_i , we split each f_i into two parts, each meeting one of the two bounds. To prove (2), note that the solution to (3) is:

$$\begin{aligned} l_i &= f_i \frac{\bar{\rho} - \rho_i}{\bar{\rho} - \underline{\rho}} \\ h_i &= f_i \frac{\rho_i - \underline{\rho}}{\bar{\rho} - \underline{\rho}}, \end{aligned}$$

Equation (2) follows, since

$$\begin{aligned} \underline{\rho}^2 l_i + \bar{\rho}^2 h_i &= \frac{f_i}{\bar{\rho} - \underline{\rho}} (\underline{\rho}^2 \bar{\rho} - \rho_i \underline{\rho}^2 + \rho_i \bar{\rho}^2 - \underline{\rho} \bar{\rho}^2) \\ &= \left(\rho_i^2 + (\bar{\rho} - \rho_i)(\rho_i - \underline{\rho}) \right) f_i \\ &\geq \rho_i^2 f_i \\ &= \frac{a_i^2}{f_i}. \end{aligned}$$

Next, note that since $\sum_{i=1}^M \rho_i f_i = \sum_{i=1}^M a_i = 1$, we get

$$\begin{aligned} \sum_{i=1}^M l_i &= \frac{\bar{\rho}}{\bar{\rho} - \underline{\rho}} \sum_{i=1}^M f_i - \frac{1}{\bar{\rho} - \underline{\rho}} \\ \sum_{i=1}^M h_i &= \frac{1}{\bar{\rho} - \underline{\rho}} - \frac{\underline{\rho}}{\bar{\rho} - \underline{\rho}} \sum_{i=1}^M f_i. \end{aligned} \quad (4)$$

Thus, using and Equations (2) and (4), we get

$$\begin{aligned} \sum_{i=1}^M \frac{a_i^2}{f_i} &\leq \sum_{i=1}^M (\underline{\rho}^2 l_i + \bar{\rho}^2 h_i) \\ &= \underline{\rho}^2 \sum_{i=1}^M l_i + \bar{\rho}^2 \sum_{i=1}^M h_i \\ &= \frac{\underline{\rho}^2 \bar{\rho}}{\bar{\rho} - \underline{\rho}} \sum_{i=1}^M f_i - \frac{\underline{\rho}^2}{\bar{\rho} - \underline{\rho}} + \frac{\bar{\rho}^2}{\bar{\rho} - \underline{\rho}} - \frac{\underline{\rho} \bar{\rho}^2}{\bar{\rho} - \underline{\rho}} \sum_{i=1}^M f_i \\ &= \underline{\rho} + \bar{\rho} - \underline{\rho}\bar{\rho} + \underline{\rho}\bar{\rho}\Delta. \quad \square \end{aligned}$$

The following corollary of Lemma 3 is for the special case of $\underline{\rho} = 1$.

Corollary 1. *Let a_1, \dots, a_M and f_1, \dots, f_M be positive real numbers such that $\sum_{i=1}^M a_i = 1$ and $\sum_{i=1}^M f_i \leq 1$. Let $\Delta = 1 - \sum_{i=1}^M \frac{a_i}{f_i}$. If $\bar{\rho}$ is such that $1 \leq \frac{a_i}{f_i} \leq \bar{\rho}$ for all i , then $\sum_{i=1}^M \frac{a_i^2}{f_i} \leq 1 + \bar{\rho}\Delta$.*

Additional Notation. All logarithms in this paper are to base 2. We use the following convenient notation, defined for all $x > 0$:

$$\begin{aligned} \llbracket x \rrbracket &\stackrel{\text{def}}{=} 2^{\lceil \log x \rceil} \\ L(x) &\stackrel{\text{def}}{=} 2^{\log x - \lfloor \log x \rfloor} = \frac{x}{\llbracket x \rrbracket} \end{aligned}$$

3 Weighted Trees, Perfectly Periodic Schedules, and Embedding

In this section, we introduce the key concept for our algorithms: the representation of some perfectly periodic schedules as trees. We explain how to derive schedules from trees, and prove some sufficient conditions for a set of periods to be embedded in a tree.

3.1 Weighted trees

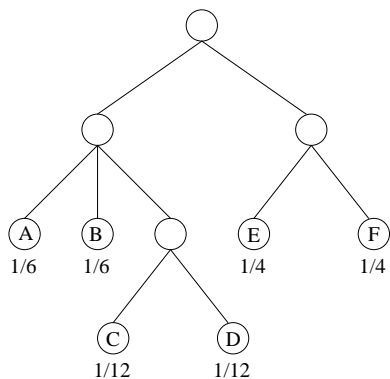


Fig. 2. Example of a weighted tree

Our algorithms use rooted trees we call *weighted trees*, where each node has a weight defined as follows. The weight of the root is 1 (unless explicitly stated otherwise), and the weight of a non-root node with parent u is the weight of u divided by the number of children of u . Figure 2 illustrates an example of a weighted tree with six leaves A, B, C, D, E, F and respective weights $\frac{1}{6}, \frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{4}, \frac{1}{4}$. In some cases, we consider subtrees of weighted trees. Subtrees can be treated as weighted trees by assigning the root a weight less than one and defining the rest of the weights as before.

We say that a sequence of n frequencies can be *embedded* in a weighted tree with at least n leaves, if there exists a 1-1 mapping that assigns to each frequency f_i a distinct leaf whose weight is f_i . The following theorem shows how to generate a perfectly periodic schedule from any given weighted tree.

Theorem 1. *If a sequence of frequencies f_1, \dots, f_n can be embedded in a weighted tree, then there exists a perfectly periodic schedule S whose frequencies are f_1, \dots, f_n .*

Proof. By induction on the height of the tree. The base case is a tree of height 0, which means a single page of frequency 1 and a trivial periodic schedule. For the induction step, suppose that the lemma holds for trees of height at most h , and consider a tree T of height $h + 1$. Let the children of the root be v_1, \dots, v_k . Consider the subtrees rooted at v_1, \dots, v_k . Their heights are at most h , and the weight of a leaf in such a subtree is k times the weight of that leaf in the original tree T . Using the assumption that f_1, \dots, f_n are embeddable in T , we have a

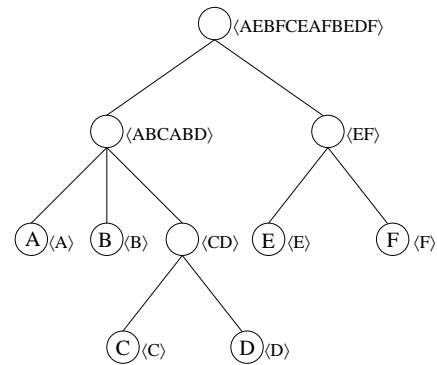


Fig. 3. Proof of Theorem 1 applied to the tree of Figure 2. The cycle of the schedule corresponding to each node is represented in brackets next to the node.

natural mapping from the frequencies to the leaves of the subtrees as well, such that each frequency f is mapped to a subtree-leaf whose weight is kf . Thus, by the induction hypothesis, there exist k perfectly periodic schedules S_1, \dots, S_k where each page i has a period of $1/kf_i$. The schedule S for T is defined by interleaving the schedules S_1, \dots, S_k in a round-robin fashion: the first k pages of S are the first pages of S_1, S_2, \dots, S_k in order; then the second pages of S_1, S_2, \dots, S_k in order, and so on (see example in Figure 3). Consider a page i that appears in schedule S_j with frequency kf_i . Since S_j is perfectly periodic, i is scheduled in S_j exactly every $1/kf_i$ time units. By construction we have that i is scheduled in S exactly every $1/f_i$ time units, proving that S is perfectly periodic with the right frequencies. \square

Remarks.

1. For the purpose of constructing schedules, we may assume, without loss of generality, that no node in a weighted tree has exactly one child: all such nodes can be eliminated.
2. If the sum of the frequencies is strictly less than 1, then there are time-slots in which no page is broadcast. This corresponds to the case in which some of the leaves are not associated with pages.
3. Note that the length of the cycle of the schedule for a given tree may be exponential in the size of the tree. The algorithms we present in the following sections will represent schedules by trees rather than full cycles.

The converse to Theorem 1 does not hold in general, as stated in the following theorem.

Theorem 2. *There exist perfectly periodic schedules whose frequencies cannot be embedded in a tree.*

Proof. Note that in a weighted tree, the degree of the root divides all periods in the corresponding schedule. In addition, we may assume w.l.o.g. that the degree of the root is greater than 1 (see Remark 1 above). The theorem is proven with a simple example that can easily be generalized. Consider the following schedule:

A B C _ _ _ A _ _ _ B A _ _ _ C A _ _ B _ _ A _ _ _ _

In this schedule, the period of Page A is 6, the period of page B is 10, the period of page C is 15, and there

are 20 other pages, each with period 30. To see that this schedule cannot be represented by a tree, suppose it can and consider the root: its degree must divide the periods 6, 10, 15, and hence it cannot be greater than 1, contradiction. \square

3.2 Embedding results

In the algorithms we develop, we use various techniques to embed a frequency sequence in a weighted tree, assuming the sequence satisfies certain properties.

3.2.1 Root degree k , other nodes have two or no children

We start by considering a tree whose root has k children, and each of its children is a root of a binary tree. This case is a variant of Kraft's inequality [12, page 47].

Lemma 4. *Let $\{f_i\}_{i=1}^n$ be a sequence of frequencies such that for all $1 \leq i \leq n$, $f_i = \frac{W}{k}2^{-r_i}$ for some real $W \leq 1$, and integers $k > 0$ and $r_i \geq 0$. If $\sum_{i=1}^n f_i \leq W$, then $\{f_i\}_{i=1}^n$ can be embedded in a tree of weight W .*

Proof. We prove the lemma with an explicit tree construction algorithm. Except for the root, all nodes in the tree will be either leaves or will have exactly two children. The initial tree is a root node with k children, each with weight of W/k . The algorithm then proceeds as follows.

1. Let F be the multiset of the f_i values.
2. If $F = \emptyset$, terminate. Otherwise, let $f = \max(F)$.
3. Let v be an unmapped leaf with minimal weight. Let w_v be its weight.
4. If $w_v = f$, then map f to v , set $F \leftarrow F \setminus \{f\}$, and go to Step 2.
5. Otherwise, add to the tree two children of v , each with half the weight of v , and go to Step 3.

It is straightforward to verify the following invariant that holds before each execution of Step 4: $f \leq w_v$, and the total weight of unmapped leaves is at least $\sum_{f_i \in F} f_i$. Success is guaranteed by the fact that all frequencies in F and all leaf weights have the form $\frac{W}{k}2^{-i}$ for some integer $i \geq 0$. Moreover, since Steps 4 and 5 strictly decrease either w_v or $\sum_{f_i \in F} f_i$, the algorithm must terminate. \square

By setting $k = 1$ in the above lemma we get for pure binary trees the following known result [5].

Corollary 2. *For $1 \leq i \leq n$, let $f_i = W2^{-r_i}$ for some integer $r_i \geq 0$. Suppose $\sum_{i=1}^n f_i \leq W$ for some $W \leq 1$. Then the sequence $\{f_i\}_{i=1}^n$ can be embedded in a sub-tree of total weight W .*

3.2.2 Root degree k , other nodes have 0, 2 or 3 children

Next, we consider a tree with root degree k , where all nodes on each leaf-root path have two children, except the root, the leaf, and possibly one node with 3 children.

Lemma 5. *Let f_1, \dots, f_n be a sequence of numbers. Suppose that there exist an integer k and integers r_1, \dots, r_n such that $f_i \in \{\frac{1}{k}2^{-r_i}, \frac{1}{3k}2^{-r_i}\}$ for all $1 \leq i \leq n$. Let $f_t = \max\{f_i \mid f_i = \frac{1}{3k}2^{-r_i}\}$. If $\sum_{i=1}^n f_i \leq 1 - 2f_t$, then f_1, \dots, f_n can be embedded in a weighted tree.*

Proof. Intuitively, the idea is to coalesce triplets and pairs of equal frequencies of the form $\frac{1}{3k}2^{-r_i}$, which allows us to apply Lemma 4. In detail, the embedding procedure works as follows. Let G denote the sequence of all current frequencies, and let $G_3 \subseteq G$ denote the sequence of all current frequencies of the form $\frac{1}{3k}2^{-r_i}$. The values of G and G_3 change throughout the algorithm.

Step 1. Merge every three equal frequencies. That is, if $g_1, g_2, g_3 \in G_3$, and $g_1 = g_2 = g_3$, we remove them from G (and therefore from G_3), and add a new frequency $g' = g_1 + g_2 + g_3$ to G (note that $g' \notin G_3$ since it does not have the form $\frac{1}{3k}2^{-r_i}$).

Step 2. Merge iteratively every two equal frequencies which are not the largest frequencies in G_3 . Formally, if $g_1, g_2 \in G_3$ and $g_1 = g_2 \neq \max(G_3)$, we remove them from G , replace them with $g' = g_1 + g_2$, and iterate.

Consider now the resulting set of frequencies, when no more merges can be applied. If $G_3 = \emptyset$ at this point, then we proceed directly to Step 4 below. Otherwise, must deal with G_3 . We start with the following observations. Let f'_t be the new maximal value in G_3 . We have the following immediate properties.

- $f'_t \leq f_t$. This is true since in Step 2 we never merge the largest frequency.
- There are at most two frequencies in G_3 with value f'_t .
- For any given value $v < f'_t$, there is at most one frequency in G_3 with value v .

It follows that

$$\sum_{g_j \in G_3} g_j < 2f'_t + f'_t \sum_{r=1}^{\infty} 2^{-r} = 3f'_t. \quad (5)$$

Therefore, we can do the following.

Step 3. Replace all frequencies in G_3 with one frequency $g_t = 3f'_t$, i.e., $G \leftarrow G \setminus G_3 \cup \{g_t\}$. By Eq. (5) this step increases the total weight in G by at most $2f'_t \leq 2f_t$. Since $\sum_{i=1}^n f_i \leq 1 - 2f_t$ by assumption, we have that after this step, $\sum_{g \in G} g \leq 1$.

Step 4. Now all weights in G are now of the form $\frac{1}{k}2^{-r_j}$. We apply Lemma 4 to get an embedding of the frequencies $g_j \in G$ in a tree T_g .

Step 5. Each $g_j \in G$ represents a set of original frequencies that were merged in previous steps. Our final step is to embed the original frequencies $\{f_i\}$ in the leaves of a weighted tree T_f by expanding each $g_j \in G$ and its corresponding leaf l_j in T_g as follows. If g_j represents a single original frequency f_i , then f_i is mapped to l_j . If g_j represents a set of original frequencies, then every f_i in g_j is of the form $\frac{1}{3k}2^{-r_i}$, and $\sum_{f_i \in g_j} f_i$ is at most g_j . In this case, by Lemma 4, the sequence $\{f_i \mid f_i \in g_j\}$ can be embedded in some weighted sub-tree T_j of total weight g_j ; we therefore replace l_j by the subtree T_j . Repeating the process for all $g_j \in G$ completes the construction. \square

3.2.3 Root degree k , and at most one other non-binary ancestor

For the embedding used in Algorithm A, we prove a result that allows us to use trees with various degrees. We start with a simple combinatorial result.

Lemma 6. *Let f_1, f_2, \dots, f_n be a sequence of real numbers such that $\sum_i f_i = K$ for some integer K , and such that for each i , $f_i = 2^{-r_i}$ for some non-negative integer r_i . Then the sequence can be partitioned to K parts, such that the sum of the numbers in each part is exactly 1.*

Proof. By induction on n . The lemma is clearly true if $n = K$, since in this case we must have $f_i = 1$ for all i . Suppose now that $n > K$. Assume w.l.o.g. that $f_i \geq f_{i+1}$ for all $1 \leq i < n$. Then we must have that $f_n < 1$. Furthermore, $f_{n-1} = f_n$, since otherwise $\sum f_i$ cannot be an integer. The lemma follows from the induction hypothesis applied to the sequence $f_1, \dots, f_{n-2}, f_{n-1} + f_n$. \square

We can now state and prove the embedding result.

Lemma 7. *Let $F = \{f_i\}$ be a sequence of positive numbers such that for all i , $f_i = \frac{1}{p_i 2^{r_i}}$ for some non-negative integers p_i, r_i . Let L, H be such that $L \leq p_i \leq H$ for all i . If*

$$\frac{L+H}{2} \cdot (H-L+1) \cdot \max(F) \leq 1 - \sum_{f_i \in F} f_i,$$

then F can be embedded in a weighted tree.

Proof. We construct a tree and a mapping as follows. Define, for a given number p , the sequence $F_p = \{f_i \mid p_i = p\}$. We first add to F_p (and hence to F) some *pseudo members* with weight $\min(F_p)$ so as to make the sum of values in F_p a multiple of $p \cdot \max(F_p)$. The pseudo members will be embedded in the tree, but will be omitted from the mapping. This process increases $\sum_{f_i \in F_p} f_i$ by at most $p \cdot \max(F_p) \leq p \cdot \max(F)$. We do the above procedure for every p_i . Since the p_i 's are distinct integers in the range $[L, H]$, their sum is at most $\frac{L+H}{2} \cdot (H-L+1)$, and hence, the total sum of F is increased by the addition of pseudo members by at most $\frac{L+H}{2} \cdot (H-L+1) \max(F)$. Therefore, the assumption on the initial sum of frequencies in F implies that after the pseudo members are added, we still have that the sum of all frequencies is not more than 1.

We now argue that each sequence F_p it can be partitioned such that the sum of numbers in each part is a power of $1/2$. To see that, consider the normalized sequence

$$\hat{F}_p = \left\{ \frac{f}{\max(F_p)} \mid f \in F_p \right\}$$

By Lemma 6, there exists an integer b such that we can partition \hat{F}_p into $b \cdot p$ parts, where the sum of the numbers in each part is 1. Viewing this partition as a partition of F_p , and merging together every p parts, we obtain a partition of F_p into b parts such that the sum of each part

is a power of $1/2$. Repeating the process for each F_p , we get a partition of F into parts such that the sum of the numbers in each part is a power of $1/2$. Using Lemma 4, we embed these parts in a binary weighted tree (where each part is mapped to a leaf). Finally, each leaf that corresponds to a part of F_p for some p , is expanded in p binary weighted trees, using Lemma 4 once again. Omitting the pseudo members from the mapping completes the construction of the tree and the mapping for F . \square

4 Algorithm C: $(\frac{4}{3} + \frac{2}{3}a_1)$ -approximation

In this section, we present a scheduling algorithm with approximation factor $\frac{4}{3} + \frac{2a_1}{3}$. The schedule constructed by this algorithm is particularly simple because all periods are powers of $1/2$.

input: demand probabilities $\{w_1 \dots w_M\}$
such that $w_i \geq w_{i+1}$ for $1 \leq i < M$

output: frequencies $\{f_i\}$

shorthand: $\Delta = \sum_{i=1}^M (a_i - f_i)$

for $i \in \{1, \dots, M\}$ **do**

$a_i \leftarrow \frac{\sqrt{w_i}}{\sum_{j=1}^M \sqrt{w_j}}$

$f_i \leftarrow \lfloor a_i \rfloor$

$\gamma \leftarrow \frac{4}{3} + \frac{2a_1}{3}$

$\Gamma \leftarrow \{i \mid L(a_i) > \gamma\}$

for $i \in \Gamma$ **do**

if $f_i \leq \Delta$

then $f_i \leftarrow 2 \cdot \lfloor a_i \rfloor$ $\Delta \geq 0$ after this step too

Fig. 4. Algorithm C: $(\frac{4}{3} + \frac{2}{3}a_1)$ -approximation.

The formal description of Algorithm C appears in Figure 4. The idea is as follows. After computing the optimal frequencies $\{a_i\}$, they are first rounded down to $\lfloor a_i \rfloor$, i.e., the nearest power of $1/2$ from below. In this rounding, each frequency a_i is multiplied by a factor of $L(a_i) \in [1, 2)$ (recall that $L(x) = x / \lfloor x \rfloor$). Consider now the “residual capacity” of the schedule, namely the fraction of un-allocated time slots (denoted Δ in the algorithm). If this fraction is small, then Lemma 3 guarantees a good bound on the approximation factor. And if this fraction is large, we can round *up* some of the frequencies, thus improving the overall approximation factor. Hence, after the initial rounding, the algorithm scans (in an arbitrary order) the frequencies which were rounded down by at least a certain factor γ , and doubles their allocated frequency, if there is enough residual capacity. Choosing $\gamma = \frac{4}{3} + \frac{2a_1}{3}$, we get that the worst case approximation is no more than γ .

We now formally prove the correctness of the algorithm and analyze its approximation factor.

Theorem 3. *There exists a perfectly periodic schedule S for the frequencies produced by Algorithm C such that $A(S) \leq \frac{4}{3} + \frac{2a_1}{3}$.*

Proof. To see that there exists a perfectly periodic schedule, note that all frequencies computed by Algorithm C are powers of $1/2$ and their sum is at most one, and hence Lemma 4 ensures that they can be embedded in a tree, which in turn can be transformed into a perfectly periodic schedule S using Theorem 1.

To bound the approximation factor, we first bound $\rho_i = \frac{a_i}{f_i}$, for each page i in the following three immediate observations. Let $\bar{T} = (\{1, \dots, M\} \setminus \Gamma) = \{i \mid L(a_i) \leq \gamma\}$.

1. If $i \in \bar{T}$, then $1 \leq \rho_i \leq \gamma$. This follows from the fact that for $i \in \bar{T}$, $\rho_i = L(a_i)$.
2. If $i \in \Gamma$ and $f_i = 2 \lfloor a_i \rfloor$, then $\frac{\gamma}{2} < \rho_i < 1$. This follows from the fact that in this case, $\rho_i = L(a_i)/2$.
3. If $i \in \Gamma$ and $f_i = \lfloor a_i \rfloor$, then $\gamma < \rho_i < 2$. Follows from the fact that in this case, $\rho_i = L(a_i)$.

Now, let $J = \{i \mid i \in \Gamma, f_i = \lfloor a_i \rfloor\}$. When the algorithm terminates, there are two possible cases. If $J = \emptyset$, then by observations 1 and 2 above, $\rho_i \leq \gamma$ for all i , and hence, in this case we have

$$A(S) \leq C(S) = \sum_{i=1}^M a_i \rho_i \leq \gamma \sum_{i=1}^M a_i = \gamma.$$

Otherwise, there exists some $i_0 \in J$. In this case, $f_{i_0} > \Delta$ by the algorithm, and hence $\Delta < \rho_{i_0} = \frac{a_{i_0}}{\gamma} \leq \frac{a_1}{\gamma}$ by Observation 3 above. In addition, we have by Observations 1, 2, and 3 that $\frac{\gamma}{2} < \rho_i < 2$ for all $i \in \{1, \dots, M\}$. Therefore, using Lemma 3, we can conclude that if $J \neq \emptyset$, then

$$\begin{aligned} A(S) &\leq \underline{\rho} + \bar{\rho} - \underline{\rho}\bar{\rho} + \underline{\rho}\bar{\rho}\Delta \\ &= \frac{\gamma}{2} + 2 - \gamma + \gamma\Delta \\ &= 2 - \frac{\gamma}{2} + a_1. \end{aligned}$$

In summary, we have that $A(S) \leq \max(\gamma, 2 - \frac{\gamma}{2} + a_1)$. Since for $\gamma = \frac{4}{3} + \frac{2a_1}{3}$ we have $\gamma = 2 - \frac{\gamma}{2} + a_1$, the result follows. \square

5 Algorithm B: $\left(\frac{9}{8-20a_1}\right)$ -approximation

In this section, we present an algorithm whose approximation factor is guaranteed to be at most $\frac{9}{8-20a_1}$. The algorithm works only for $a_1 < \frac{1}{42}$. This algorithm, although fairly simple in terms of the final code, is quite complicated to analyze. We first present the algorithm that computes the frequencies, then show that these frequencies can be embedded in a tree (thus getting a perfectly periodic schedule), and finally analyze the approximation factor of these frequencies.

5.1 Description

Algorithm B generates a tree whose root degree is either 4, 5, 6 or 7, and all other non-leaf nodes have 2 or 3 children. The algorithm calculates for each page a frequency, such that on one hand, a tree embedding is

possible using Lemma 5 (which requires sufficiently large residual capacity Δ), and on the other hand, a small approximation factor can be guaranteed, using Lemma 3 (which depends on sufficiently small Δ).

Informally, the first step of the algorithm is to choose the value of the root degree k : this choice is quite subtle, and we describe it later in this section. After determining k , the algorithm makes an initial frequency assignment to each page, by rounding down a_i to the nearest value of one of the forms $\frac{1}{k}2^{-r_i}$ or $\frac{1}{3k}2^{-r_i}$. The choice of k guarantees that after this step, $\Delta \geq 2f_t$, for $f_t = \max\{f_i \mid f_i = \frac{1}{3k}2^{-r_i}\}$, and hence these frequencies can be embedded in a tree by Lemma 5. Then the algorithm performs an ‘‘enhancement’’ step. In this step, pages whose frequencies were rounded down by a sufficiently large factor γ (where $\gamma \approx 9/8$) are rounded up, provided that the residual capacity remains large enough to ensure that Lemma 5 can still be applied.

Pseudo-code for the algorithm is given in Figure 5. Recall that $\lfloor x \rfloor = 2^{\lfloor \log x \rfloor}$ and $L(x) = \frac{x}{\lfloor x \rfloor}$ for any number $x > 0$.

input: demand probabilities $\{w_1 \dots w_M\}$
such that $w_i \geq w_{i+1}$ for $1 \leq i < M$

output: frequencies $\{f_i\}$

shorthand: $\Delta = \sum_{i=1}^M (a_i - f_i)$

for $1 \in \{1, \dots, M\}$ **do**

$a_i \leftarrow \frac{\sqrt{w_i}}{\sum_{i=1}^M \sqrt{w_i}}$

choose k as described in Section 5.2

for $1 \in \{1, \dots, M\}$ **do**

$p_i \leftarrow \frac{\lfloor a_i k \rfloor}{k}$ *by definitions, $p_i = \frac{a_i}{L(a_i k)}$*

$\gamma \leftarrow \frac{9}{8-20p_1}$

$\Gamma_1 \leftarrow \{i \mid 1 \leq L(a_i k) \leq \gamma\}$

$\Gamma_2 \leftarrow \{i \mid \gamma < L(a_i k) < \frac{4}{3}\}$

$\Gamma_3 \leftarrow \{i \mid \frac{4}{3} \leq L(a_i k) \leq \frac{4}{3}\gamma\}$

$\Gamma_4 \leftarrow \{i \mid \frac{4}{3}\gamma < L(a_i k) < 2\}$

for $1 \in \{1, \dots, M\}$ **do** *Initial assignment*

$f_i \leftarrow \begin{cases} p_i, & \text{if } i \in \{\Gamma_1, \Gamma_2\} \\ \frac{4}{3}p_i, & \text{if } i \in \{\Gamma_3, \Gamma_4\} \end{cases}$ *now $f_i \leq a_i$ for all i*

for $1 \in \{1, \dots, M\}$ **do** *Enhancement step*

if $i \in \Gamma_2$ and $\Delta \geq \frac{8}{3}p_1 + \frac{1}{3}p_i$ **then** $f_i \leftarrow \frac{4}{3}p_i$

else if $i \in \Gamma_4$ and $\Delta \geq \frac{8}{3}p_1 + \frac{2}{3}p_i$

then $f_i \leftarrow 2p_i$ *now $f_i \in \{p_i, \frac{4}{3}p_i, 2p_i\}$ for all i*

Fig. 5. Algorithm B: $\left(\frac{9}{8-20a_1}\right)$ -approximation.

5.2 Choosing the root degree k : Procedure and analysis

We now describe how to choose k . We start with the following definition.

$$Q(j) = \left\{i \mid \frac{8}{4+j} \leq L(a_i) < \frac{8}{3+j}\right\} \text{ for } j = 1, 2, 3, 4 \quad (6)$$

Let $j_0 \in \{1, 2, 3, 4\}$ be such that $\sum_{i \in Q(j_0)} a_i$ is maximized. Note that since $Q(1), \dots, Q(4)$ form a partition of the a_i sequence, this sum is at least $1/4$. We choose

as the root degree $k \leftarrow j_0 + 3$. To show the significance of this choice we use the following definition.

$$Q'(k) = \{i \mid \frac{8}{5} \leq L(ka_i) < 2\} \text{ for } k \geq 1 \quad (7)$$

The following lemma clarifies the relation between the $Q(j)$ and the $Q'(k)$ sets.

Lemma 8. *If $j \in \{1, 2, 3, 4\}$ and $k = j + 3$, then $Q'(k) \supseteq Q(j)$.*

Proof. Let $j \in \{1, 2, 3, 4\}$. The key to the proof is the following claim:

$$L(ka_i) = L(k)L(a_i) \text{ for all } i \in Q(j). \quad (8)$$

To prove (8), first observe that the following inequalities are equivalent:

$$\begin{aligned} \frac{8}{4+j} \llbracket a_i \rrbracket &\leq a_i < \frac{8}{3+j} \llbracket a_i \rrbracket \\ \frac{3+j}{4+j} \llbracket a_i \rrbracket &\leq \frac{ka_i}{8} < \llbracket a_i \rrbracket \\ \log \frac{3+j}{4+j} + \lceil \log a_i \rceil &\leq \log \frac{ka_i}{8} < \lceil \log a_i \rceil. \end{aligned}$$

Since $j > 0$, it follows from integrality that

$$-1 + \lceil \log a_i \rceil \leq \lceil \log \frac{ka_i}{8} \rceil < \lceil \log a_i \rceil,$$

and hence $\lceil \log(ka_i) \rceil = \lceil \log a_i \rceil + 2$. On the other hand, note that since $4 \leq k < 8$, we have that $L(k) = \frac{k}{4}$ by definition, and therefore

$$L(ka_i) = \frac{ka_i}{2^{\lceil \log(ka_i) \rceil}} = \frac{ka_i}{4 \cdot \llbracket a_i \rrbracket} = L(k)L(a_i).$$

This establishes Eq. (8). Now, to prove the lemma, let $i \in Q(j)$. Then $L(ka_i) = \frac{3+j}{4} \cdot L(a_i)$ by Eq. (8). Hence, by definition of $Q(j)$,

$$\frac{8}{5} \leq \frac{3+j}{4} \cdot \frac{8}{4+j} \leq L(ka_i) < \frac{3+j}{4} \cdot \frac{8}{3+j} = 2.$$

The lemma now follows from the definition of $Q'(k)$. \square

Since $\sum_{i \in Q(k-3)} a_i \geq \frac{1}{4}$, the choice of k and Lemma 8 imply the following corollary.

Corollary 3. *For the choice of $k = j_0 + 3$ described above, we have $\sum_{i \in Q'(k)} a_i \geq 1/4$.*

5.3 Analysis

We first prove that the frequency sequence generated by Algorithm B has a perfectly periodic schedule. We need the following lemma (this is the only place where we use the assumption that $a_i < \frac{1}{42}$).

Lemma 9. *Suppose that $a_1 < \frac{1}{42}$. Then after the initial assignment, $f_i \leq \frac{1}{48}$ for all $i \in \Gamma_3 \cup \Gamma_4$ for any $k \in \{4, 5, 6, 7\}$.*

Proof. We proceed by case analysis, depending on the value of k .

$k = 4$:

$$f_i = \frac{4 \llbracket 4 \cdot a_i \rrbracket}{3 \cdot 4} = \frac{4 \llbracket a_i \rrbracket}{3} \leq \frac{4}{3 \cdot 64} = \frac{1}{48}.$$

$k = 5$:

$$f_i = \frac{4 \llbracket 5 \cdot a_i \rrbracket}{3 \cdot 5} \leq \frac{4}{15} \cdot \llbracket 5/42 \rrbracket = \frac{4}{15} \cdot \frac{1}{16} = \frac{1}{60}.$$

$k = 6, 7$: We first note that after the initial assignment, $f_i \leq a_i$ for all i , and hence we need only to consider the case $\frac{1}{42} < a_i \leq \frac{1}{48}$ (this observation holds also when $k \in \{4, 5\}$, but we managed without it). We claim that for $k \in \{6, 7\}$ and $a_i \in (\frac{1}{42}, \frac{1}{48}]$, it must be the case that $i \notin \Gamma_3 \cup \Gamma_4$: this is because for the values of a_i and k in the case we consider, we can bound $a_i k$ by

$$\frac{1}{8} = \frac{6}{48} \leq a_i k < \frac{7}{42} = \frac{1}{6},$$

and hence $1 \leq L(a_i k) < \frac{4}{3}$, i.e., $i \in \Gamma_1 \cup \Gamma_2$ and we are done. \square

Theorem 4. *If $a_1 < \frac{1}{42}$, then there exists a perfectly periodic schedule for the frequencies produced by Algorithm B.*

Proof. Let $f_t = \max \{f_i \mid f_i = \frac{1}{3k} 2^{-r_i}\}$. We first claim that after the initial assignment step, the following holds:

$$\Delta \geq 2f_t. \quad (9)$$

Consider $Q'(k)$ as defined in Eq. (7). To prove Eq. (9), note that since $f_i \leq a_i$ for all i after the initial assignment and $Q'(k)$ is a subset of the frequencies, it suffices to show that $\sum_{i \in Q'(k)} (a_i - f_i) \geq 2f_t$. By definition of $Q'(k)$, we have that $L(a_i k) \geq \frac{8}{5}$ for all $i \in Q'(k)$. It follows that in the initial assignment, the algorithm sets $f_i = \frac{4}{3} p_i$ for all $i \in Q'(k)$. Since $L(a_i k) = \frac{a_i}{p_i}$ by definition, we get that

$$\frac{a_i}{f_i} = \frac{a_i/p_i}{f_i/p_i} \geq \frac{8/5}{4/3} = \frac{6}{5}. \quad (10)$$

In summary, we get that

$$\begin{aligned} \Delta &= \sum_{i=1}^M (a_i - f_i) \\ &\geq \sum_{i \in Q'(k)} (a_i - f_i) \\ &\geq \frac{1}{6} \sum_{i \in Q'(k)} a_i && \text{by Eq. (10)} \\ &\geq \frac{1}{24}. && \text{by Cor. 3} \end{aligned}$$

On the other hand, by Lemma 9, we have that $f_t \leq \frac{1}{48}$, so we may conclude that Eq. (9) holds true after the initial assignment step.

We now proceed to show that Eq. (9) holds also after the enhancement step. To see that, note that in the enhancement step, the algorithm maintains the invariant $\Delta \geq \frac{8}{3}p_1$. Also note that $\frac{f_i}{p_i} = \frac{4}{3}$ for every f_i of form $\frac{1}{3k}2^{-r_i}$, and hence, after the enhancement step we have $\Delta \geq \frac{8}{3}p_1 \geq \frac{8}{3}p_i = 2f_i$ and Eq. (9) still holds true.

To conclude, the set of frequencies computed by the algorithm is such that every frequency is of the form $f_i = \frac{1}{k}2^{-r_i}$ or $f_i = \frac{1}{3k}2^{-r_i}$, and that $\Delta \geq 2f_i$. Therefore, by Lemma 5, the sequence $\{f_i\}$ can be embedded in a weighted tree, and by Theorem 1 there exists a perfectly periodic schedule with frequencies $\{f_i\}$. \square

We now turn to analyze the approximation factor of algorithm B.

Theorem 5. *Let S be a perfectly periodic schedule whose frequencies are generated by Algorithm B. Then $A(S) \leq \frac{9}{8-20a_1}$.*

Proof. Recall that $\rho_i = \frac{a_i}{f_i}$. We use the following straightforward claims.

1. *If $i \in \Gamma_1 \cup \Gamma_3$, then $1 \leq \rho_i \leq \gamma$.*
To see that, consider $i \in \Gamma_1$. By the algorithm, $f_i = p_i$. Clearly, $f_i \leq a_i$ hence $\rho_i \geq 1$. Also, by definition of Γ_1 , we have $\frac{a_i}{p_i} \leq \gamma$ and hence $\rho_i \leq \gamma$. The proof for Γ_3 is similar.
2. *If $i \in \Gamma_2$ and $f_i = \frac{4}{3}p_i$, then $\frac{3}{4}\gamma < \rho_i < 1$; also, if $i \in \Gamma_4$ and $f_i = 2p_i$, then $\frac{2}{3}\gamma < \rho_i < 1$.*
To see that this claim is true, consider $i \in \Gamma_2$ such that $f_i = \frac{4}{3}p_i$. By definition of Γ_2 , $\gamma < \frac{a_i}{p_i} < \frac{4}{3}$, and hence $\frac{3}{4}\gamma < \rho_i < 1$. Similarly, consider $i \in \Gamma_4$ such that $f_i = 2p_i$. By definition of Γ_4 , $\frac{4}{3}\gamma < \frac{a_i}{p_i} < 2$, and hence $\frac{2}{3}\gamma < \rho_i < 1$.
3. *If $i \in \Gamma_2$ and $f_i = p_i$, the $\gamma < \rho_i < \frac{4}{3}$; also, if $i \in \Gamma_4$ and $f_i = \frac{4}{3}p_i$, then $\gamma < \rho_i < \frac{3}{2}$.*
To see that this claim is true, consider $i \in \Gamma_2$ such that $f_i = p_i$. By definition of Γ_2 , $\gamma < \frac{a_i}{p_i} < \frac{4}{3}$, and hence $\gamma < \rho_i < \frac{4}{3}$. Similarly, consider $i \in \Gamma_4$ such that $f_i = \frac{4}{3}p_i$. By definition of Γ_4 , $\frac{4}{3}\gamma < \frac{a_i}{p_i} < 2$, and hence $\gamma < \rho_i < \frac{3}{2}$.

Now let $J = \{i \mid i \in \Gamma_2, f_i = p_i\} \cup \{i \mid i \in \Gamma_4, f_i = \frac{4}{3}p_i\}$. If $J = \emptyset$ (i.e., all pages in Γ_2 and Γ_4 were increased in the enhancement step), then by Claims 1 and 2, for every page $\rho_i \leq \gamma$, and hence

$$A(S) \leq \gamma.$$

Otherwise, $J \neq \emptyset$. In this case, there exists $i_0 \in J$, such that $\Delta < \frac{8}{3}p_1 + \frac{2}{3}p_{i_0}$, since otherwise the algorithm would have increased f_{i_0} in the enhancement step. This means that $\Delta \leq \frac{10}{3}p_1$ since $p_{i_0} \leq p_1$. In addition, by Claims 1, 2 and 3, we have that for all i , $\frac{2}{3}\gamma \leq \rho_i \leq \frac{3}{2}$. Consequently, from Lemma 3 we get

$$A(S) \leq \frac{3}{2} - \frac{\gamma}{3} + \frac{10}{3}\gamma p_1.$$

In summary, we have that $A(S) \leq \max(\gamma, \frac{3}{2} - \frac{\gamma}{3} + \frac{10}{3}\gamma p_1)$. For $\gamma = \frac{9}{8-20p_1}$, we get

$$A(S) \leq \gamma \leq \frac{9}{8-20a_1}. \quad \square$$

6 Algorithm A: $(1 + \frac{(3a_1)^{1/3}}{1-(3a_1)^{1/3}})$ -approximation

In this section we present our best algorithm for small values of a_1 . The algorithm works under the assumption that $a_1 < \frac{1}{3}$. The goal is to generalize the binary tree construction and to use the embedding of Lemma 7. Informally, the algorithm works as follows. First, the algorithm rounds down each a_i and sets f_i to be the maximal $\frac{1}{p_i}2^{r_i} \leq a_i$ where p_i and r_i are integers, and $x \leq p_i < 2x$ for some integer $x \geq 1$ to be determined later. This rounding guarantees that $\frac{x}{x+1}a_i \leq f_i \leq a_i$ for all i . To apply Lemma 7, we need to have sufficiently large residual capacity Δ . If Δ is too small, we did not lose much so far, and we may continue reducing the allocated frequencies f_i . But now, we round the f_i 's down to $\lfloor f_i \rfloor$. This is done until Δ is large enough.

input: demand probabilities $\{w_1 \dots w_M\}$
such that $w_i \geq w_{i+1}$ for $1 \leq i < M$

output: frequencies $\{f_i\}$

shorthand: $\Delta = \sum(a_i - f_i)$

for $i \in \{1, \dots, M\}$ **do**

$a_i \leftarrow \frac{\sqrt{w_i}}{\sum_{j=1}^M \sqrt{w_j}}$

$k \leftarrow \lfloor -\frac{\log(3a_1)}{3} \rfloor$

$x \leftarrow 2^k$

for $i \in \{1, \dots, M\}$ **do** *first phase*

$p_i \leftarrow \lceil x \cdot 2^{\lceil \log a_i \rceil} \rceil$ *note that $x \leq p_i < 2x$*

$f_i \leftarrow \frac{x}{p_i} \cdot 2^{\lceil \log a_i \rceil}$

for $1 \in \{1, \dots, M\}$ **do** *second phase*

if $\Delta < \frac{3x(x-1)}{2} \cdot \max\{f_i\}$

then $f_i \leftarrow \lfloor a_i \rfloor$

Fig. 6. Algorithm A: $(1 + \frac{(3a_1)^{1/3}}{1-(3a_1)^{1/3}})$ -approximation.

The formal description of Algorithm A appears in Figure 6. The following theorem shows that it produces a perfectly periodic schedule.

Theorem 6. *If $a_1 < \frac{1}{3}$, then there exists a perfectly periodic schedule for the frequencies produced by Algorithm A.*

Proof. Initially, we set $x = \lfloor (3a_1)^{-\frac{1}{3}} \rfloor$ and therefore $x \geq 1$ for $0 < a_1 < \frac{1}{3}$. In the first phase, for $1 \leq i \leq M$, we set f_i to be the maximal number not larger than a_i that has the form $\frac{1}{p_i}2^{r_i}$, where $x \leq p_i < 2x$ and both p_i and r_i are integers. More formally, p_i and r_i satisfy the

following inequalities:

$$\begin{aligned} \frac{1}{x2^{r_i+1}} &< a_i \leq \frac{1}{x2^{r_i}} \\ \frac{1}{p_i2^{r_i}} &\leq a_i < \frac{1}{(p_i-1)2^{r_i}} \end{aligned} \quad (11)$$

It is not difficult to verify that the values of r_i and p_i are given by the following expressions.

$$r_i = \left\lceil \log \left(\frac{1}{a_i x} \right) \right\rceil \quad (12)$$

$$p_i = \left\lceil \frac{1}{2^{r_i} a_i} \right\rceil = \left\lceil x \cdot \frac{2^{\lceil \log a_i \rceil}}{a_i} \right\rceil \quad (13)$$

(The last equality follows from the fact that x is a power of 2.) In the second phase, we iterate through the f_i 's in any order, and round them down to $\lfloor f_i \rfloor$ until $\Delta \geq \frac{(3x-1)x}{2} \cdot \max \{f_i\}$, or until all frequencies are powers of $\frac{1}{2}$. In the latter case, we can embed the frequencies in a binary tree by Lemma 4. In the former case, we can embed the frequencies in a tree using Lemma 7. To see this, note that p_i is an integral number between x and $2x-1$, i.e., in the terminology of Lemma 7, we have $L = x$ and $H = 2x - 1$, and hence $\frac{L+H}{2}(H-L+1) \leq \frac{(3x-1)x}{2}$ and the lemma is applicable. Finally, the existence of a perfectly periodic schedule follows from Theorem 1. \square

The next theorem bounds the approximation factor of Algorithm A.

Theorem 7. *If S is a schedule produced by Algorithm A, then $A(S) \leq 1 + \frac{(3a_1)^{\frac{1}{3}}}{1 - (3a_1)^{\frac{1}{3}}}$.*

Proof. We consider two cases. Suppose first that after the first phase, $\Delta \geq \frac{(3x-1)x}{2} \cdot \max \{f_i\}$. We claim that in this case $A(S) \leq 1 + \frac{1}{x}$. To see this, first note that for all i , $\frac{a_i}{f_i} \leq \frac{p_i+1}{p_i}$ by Eq. (11), and the fact that $r_i \geq 0$ by Eq. (12). Hence $A(S) \leq 1 + \frac{1}{x}$ by the fact that $\frac{p_i+1}{p_i} \leq \frac{x+1}{x}$, since $p_i \geq x$ by Eq. (13).

The second case is where $\Delta < \frac{(3x-1)x}{2} \cdot \max \{f_i\}$ after the first phase. We first bound Δ . Consider the last rounding in the second phase. Before that last rounding, $\Delta < \frac{(3x-1)x}{2} \cdot \max \{f_i\}$. The last rounding increases Δ by at most $\frac{1}{2} \max \{f_i\}$, and hence, when the second phase is over, we have

$$\Delta \leq \left(\frac{(3x-1)x}{2} + \frac{1}{2} \right) \cdot \max \{f_i\} \leq \frac{3x^2 a_1}{2}.$$

The last inequality is true since $x \geq 1$. Next, we bound $\rho_i = \frac{a_i}{f_i}$ for any $i \in \{1, \dots, M\}$. Clearly, $f_i \leq a_i$ always, and hence, $\rho_i \geq 1$. Also, $\rho_i < 2$, since $f_i > \frac{a_i}{2}$ in both the first and the second phases. Using these bounds on Δ and ρ_i we get from Corollary 1 that in this case,

$$A(S) \leq 1 + 2\Delta \leq 1 + 3x^2 a_1.$$

In summary, we have that in any case,

$$A(S) \leq \max \left\{ 1 + \frac{1}{x}, 1 + 3x^2 a_1 \right\}.$$

By the definition of x , we get that $3x^2 a_1 \leq \frac{1}{x}$. We therefore conclude that

$$A(S) \leq 1 + \frac{1}{x} \leq 1 + \frac{1}{\frac{1}{(3a_1)^{\frac{1}{3}}} - 1} = 1 + \frac{(3a_1)^{\frac{1}{3}}}{1 - (3a_1)^{\frac{1}{3}}}. \quad \square$$

7 Lower Bounds

Our upper bounds depend on $a_1 = \frac{\sqrt{\max_{i=1}^M \{w_i\}}}{\sum_{i=1}^M \sqrt{w_i}}$. These upper bounds use the lower bound on the cost of an optimal schedule (Lemma 1), derived from the relaxed problem. In this section, we show that the dependency on a_1 is unavoidable with this method: we prove that there are schedules for which the cost is at least $(1 + a_1^2)$ times the cost of the relaxed problem. Specifically, we show that for any $M \geq 2$, there exists a sequence with $(1/(M-1)) < a_1 < (1/M)$ such that the cost of any perfectly periodic schedule for the sequence is at least $(1 + a_1^2)$ times the cost of the best relaxed solution. Furthermore, for schedules that are generated by weighted trees (as are all the schedules in this paper), the cost is at least $(1 + 0.8a_1)$ times the cost of the relaxed problem. Note that the values of a_1 and M are related. Similar sequences can be constructed for other relations between these two parameters.

We start with the general result. Given $M \geq 2$ and $\varepsilon > 0$, define a set \mathcal{A} of M pages with demand probabilities $\left\{ \frac{1-\varepsilon}{M-1}, \dots, \frac{1-\varepsilon}{M-1}, \varepsilon \right\}$. For this set, we have the following result.

Theorem 8. *For any $M \geq 2$ there exists a frequency sequence in which $\frac{1}{M} < a_1 < \frac{1}{M-1}$ and such that the cost of any perfectly periodic schedule S is at least $1 + \frac{1}{(M-1)^2} > 1 + a_1^2$ times larger than the cost of the optimal relaxed cost.*

Proof. The sequence is \mathcal{A} for a small enough ε . Let S be any perfectly periodic schedule with periods $\tau_1, \tau_2, \dots \leq \tau_M$. Without loss of generality $\tau_1 \leq \tau_2 \leq \dots \leq \tau_M < \infty$ since for optimizing the objective function the period of the last page should be the largest and the rest of the demand probabilities are identical. We henceforth ignore the allocation for the last page in bounding the cost of S , except for insisting that it exists. That is, $\sum_{i=1}^{M-1} \frac{1}{\tau_i} = 1 - \delta$ for some $\delta > 0$. Consider $C(S)$. By Lemma 2, $C(S) = \sum_{i=1}^M (\tau_i a_i^2)$. Since the a_i 's of all of the first $M-1$ pages are equal, their total contribution to $C(S)$ depends only on $\sum_{i=1}^{M-1} \tau_i$. Thus $C(S)$ is minimized when $\sum_{i=1}^{M-1} \tau_i$ is minimized, subject to the constraint that $\sum_{i=1}^{M-1} \frac{1}{\tau_i} = 1 - \delta$. If the τ_i 's can get any real value, standard calculus shows that the minimum is attained for $\tau_1 = \tau_2 = \dots = \tau_{M-1} = \frac{M-1}{1-\delta}$. This value is greater than $M-1$ and less than M . But the τ_i 's must be integers and therefore, for ε sufficiently small, the minimum is attained when the τ_i 's are "almost" equal. More precisely, the minimum is at least the value attained when $\tau_1 = \tau_2 = \dots = \tau_{M-2} = M-1$ and $\tau_{M-1} = M$.

Finally, since for vanishing ε , $a_i = \frac{1}{M-1}$ for $1 \leq i \leq M-2$, we get

$$\begin{aligned} C(S) &= \sum_{i=1}^M \tau_i a_i^2 \\ &> \sum_{i=1}^{M-1} \tau_i a_i^2 \\ &\geq \frac{(M-2)(M-1) + M}{(M-1)^2} \\ &= 1 + \frac{1}{(M-1)^2}. \quad \square \end{aligned}$$

For schedules that can be embedded in trees, we prove a stronger bound, based on the arguments of Theorem 8 and the additional observation that in this case, we may assume that the two minimal frequencies are equal.

We start with the following lemma, that allows us to ignore trees with un-assigned leaves.

Lemma 10. *Let \mathcal{P} be a set of pages, and let S be a schedule for \mathcal{P} with periods $\{\tau_i\}$ such that the frequencies $\left\{\frac{1}{\tau_i} \mid i \in \mathcal{P}\right\}$ can be embedded in a tree. Then there exists a schedule S' with periods $\{\tau'_i \mid i \in \mathcal{P}\}$ such that $\sum_{i \in \mathcal{P}} \frac{1}{\tau'_i} = 1$ and $C(S') \leq C(S)$.*

Proof. If $\sum_{i \in \mathcal{P}} \frac{1}{\tau_i} = 1$ we are done. Otherwise, consider the tree T corresponding to S . Since $\sum_{i \in \mathcal{P}} \frac{1}{\tau_i} < 1$, there exist leaves in T without corresponding pages. We remove all these leaves, and contract nodes with one child, getting another tree T' . We carry over the correspondence between \mathcal{P} and the leaves of T to the leaves of T' . By Theorem 1, there exists a schedule S' corresponding to T' . Let the period of page i in S' be τ'_i , for all $i \in \mathcal{P}$. Since all leaves of T' are matched with pages in \mathcal{P} , we have that $\sum_{i \in \mathcal{P}} \frac{1}{\tau'_i} = 1$. Moreover, by construction we have that $\tau'_i \leq \tau_i$ for all i ; it follows from Lemma 2 that $C(S') \leq C(S)$. \square

The following corollary says that if a schedule can be embedded in a tree, then there are at least two pages with the minimum frequency.

Corollary 4. *Let \mathcal{P} be a set of pages, and let S be a schedule for \mathcal{P} with periods $\{\tau_i\}$ such that the frequencies $\left\{\frac{1}{\tau_i} \mid i \in \mathcal{P}\right\}$ can be embedded in a tree. Then there exists a schedule S' with periods $\{\tau'_i \mid i \in \mathcal{P}\}$ such that $C(S') \leq C(S)$ and two distinct pages $i_0, i_1 \in \mathcal{P}$ such that $\tau'_{i_0} = \tau'_{i_1} = \max\{\tau'_i \mid i \in \mathcal{P}\}$.*

Proof. Follows from Lemma 10 and the fact that without loss of generality, each leaf in the tree has a sibling. \square

We now state the result for trees. The main argument is as in Theorem 8, boosted by Corollary 4.

Theorem 9. *For any $M \geq 2$ there exists a frequency sequence in which $\frac{1}{M} < a_1 < \frac{1}{M-1}$ and such that the cost of any perfectly periodic schedule S that can be embedded in a tree, $C(S) \geq 1 + \frac{2\sqrt{2}-2}{(M-1)} > 1 + 0.8a_1$.*

Proof. The sequence is \mathcal{A} for a small enough ε . Let S be any perfectly periodic schedule with periods $\tau_1 \leq \tau_2 \leq \dots \leq \tau_M < \infty$. By Corollary 4, we may assume without loss of generality that $\tau_M = \tau_{M-1}$. Next, observe that $\tau_M \geq M$, since otherwise $\sum_{i=1}^M \frac{1}{\tau_i} > 1$. Denote $\tau_M = \tau_{M-1} = \ell$. With this notation, we have that

$$\sum_{i=1}^{M-2} \frac{1}{\tau_i} = 1 - \frac{2}{\ell}. \quad (14)$$

Now, by Lemma 2, $C(S) = \sum_{i=1}^M (\tau_i a_i^2)$. Since the a_i 's of all first $M-1$ pages are equal, their total contribution to $C(S)$ depends only on $\sum_{i=1}^{M-1} \tau_i$. We focus on the first $M-2$ terms. Using standard techniques, it can be seen that $\sum_{i=1}^{M-2} \tau_i$ is minimized subject to the constraint Eq. (14) when

$$\tau_1 = \tau_2 = \dots = \tau_{M-2} = \frac{M-2}{1 - \frac{2}{\ell}} = \frac{(M-2)\ell}{\ell-2}.$$

Since for vanishing ε we have that $a_i = \frac{1}{M-1}$ for $i = 1, \dots, M-1$, it follows that

$$\begin{aligned} C(S) &= \sum_{i=1}^M \tau_i a_i^2 \\ &> \sum_{i=1}^{M-1} \tau_i a_i^2 \\ &\geq \frac{1}{(M-1)^2} \left((M-2) \cdot \frac{(M-2)\ell}{\ell-2} + \ell \right). \end{aligned}$$

For $\ell \geq M$, the right hand side of the above inequality is minimized when $\ell = 2 + \sqrt{2}(M-2)$. With some algebraic manipulations we get that

$$\begin{aligned} C(S) &\geq \frac{((M-1) + (\sqrt{2}-1))^2}{(M-1)^2} \\ &> 1 + \frac{2\sqrt{2}-2}{M-1} \\ &\geq 1 + \frac{0.828427}{M-1}. \quad \square \end{aligned}$$

8 Conclusion

In this paper, we introduced and explored the natural concept of perfectly periodic schedules. We also introduced a novel tree methodology for designing such schedules. We have demonstrated, by algorithms and lower bounds, the significance of the quantity a_1 for perfectly periodic schedules in general, and tree-based perfectly periodic schedules in particular. There are many interesting questions we leave open. Let us list a few.

- Trees represent hierarchical schedules and do not represent all possible perfectly periodic schedules (see Theorem 2). Are there more effective ways to design perfectly periodic schedules?
- Are there better schedules than the ones we presented? Are there better lower bounds?

- In our model, the length of each page was a unit. How does one extend the algorithms to the case where each page has an arbitrary positive length?
- In our model, only one page can be served in each time unit. How does one extend the model to the case of “parallel servers,” where k pages can be served in the same time slots, for some fixed parameter k ?
- The hardness result of [8] is proved when the sum of requested frequencies is less than 1. It is not readily clear that it holds when the sum of requested frequencies is exactly 1.

Acknowledgements. We thank Rafael Hassin for helpful discussions, Jim Anderson for pointing to us the work on Pfair schedules, and the anonymous referees for their help in improving the final version of the paper.

References

1. S. Acharya, R. Alonso, M. J. Franklin, and S. B. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proc. 1995 ACM SIGMOD*, pages 199–210, 1995.
2. S. Acharya, M. Franklin, and S. Zdonik. Dissemination based data delivery using broadcast disks. *IEEE Personal Communication*, pages 50–60, Dec. 1995.
3. M. H. Ammar and J. W. Wong. The design of teletext broadcast cycles. *Performance Evaluation*, 5(4):235–242, Dec 1985.
4. M. H. Ammar and J. W. Wong. On the optimality of cyclic transmission in teletext systems. *IEEE Trans. Communication*, COM-35(11):1159–1170, Nov 1987.
5. S. Anily, C. A. Glass, and R. Hassin. The scheduling of maintenance service. *Discrete Applied Mathematics*, 82:27–42, 1998.
6. S. Anily, C. A. Glass, and R. Hassin. Scheduling of maintenance services to three machines. *Annals of Operations Research*, 86:375–391, 1999.
7. A. Bar-Noy, B. Patt-Shamir, and I. Ziper. Broadcast disks with polynomial cost functions. In *Proc. 19th INFOCOM*, Mar. 2000.
8. A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation cost of periodic scheduling. In *Proc. of the 9th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 11–20, 1998.
9. S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1996.
10. M. Y. Chan and F. Chin. Schedulers for larger classes of pinwheel instances. *Algorithmica*, 9:425–462, 1993.
11. Y. D. Chung and M.-H. Kim. QEM: A scheduling method for wireless broadcast data. In *Proc. Sixth International Conf. on Database Systems for Advanced Applications*, pages 135–142. IEEE Computer Society, 1999.
12. Robert G. Gallager. *Information Theory and Reliable Communication*. John Wiley & Sons, New York, NY, USA, 1968.
13. G. Hadley and T. M. Whitin. *Analysis of inventory systems*. Prentice-Hall, 1963.
14. R. Hassin and N. Megiddo. Exact computation of optimal inventory policies over an unbounded horizon. *Mathematics of Operations Research*, 80:534–546, 1991.
15. M. Hofri and Z. Rosberg. Packet delay under the golden ratio weighted tdm policy in a multiple-access channel. *IEEE Trans. Information Theory*, 11-33:341–349, May 1987.
16. R. Holte, L. Rosier, I. Tulchinsky, and D. Varvel. Pinwheel scheduling with two distinct numbers. *Theoretical Computer Science*, 100:105–135, 1992.
17. T. Imielinski, S. Viswanathan, and B. R. Badrinath. Energy efficient indexing on air. In R. T. Snodgrass and M. Winslett, editors, *Proc. 1994 ACM SIGMOD*, pages 25–36. ACM Press, 1994.
18. C. Kenyon and N. Schabanel. The data broadcast problem with non-uniform transmission times. In *Proc. 10th SODA*, pages 547–556, Jan 1999.
19. C. Kenyon, N. Schabanel, and N. Young. Polynomial-time approximation scheme for data broadcast. In *Proc. 32nd STOC*, pages 659–666, May 2000.
20. S. Khanna and S. Zhou. On indexed data broadcast. In *Proc. 30th ACM STOC*, pages 463–472, 1998.
21. C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.
22. A. Nisgav. Nearly optimal perfectly periodic schedules. Master’s thesis, Dept. of Electrical Engineering, Tel-Aviv University, Sept. 2000.
23. R. Roundy. 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailer systems. *Management Science*, 31:1416–1460, 1985.
24. C. J. Su and L. Tassiulas. Broadcast scheduling for information distribution. In *Proc. 16th INFOCOM*, pages 109–117. IEEE, Mar 1997.
25. R. Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32:323–330, 1980.
26. N. Vaidya and S. Hamid. Log time algorithms for scheduling single and multiple channel data broadcast. In *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM’97)*, pages 90–99, 1997.
27. W. Wei and C. Liu. On a periodic maintenance problem. *Operations Research Letters*, 2:90–93, 1983.