Improved Approximation Algorithms for Minimum-Space Advertisement Scheduling*

Brian C. Dean and Michel X. Goemans

M.I.T., Cambridge, MA 02139, USA,
bdean@theory.lcs.mit.edu, goemans@math.mit.edu

Abstract. We study a scheduling problem involving the optimal placement of advertisement images in a shared space over time. The problem is a generalization of the classical scheduling problem $P||C_{max}$, and involves scheduling each job on a specified number of parallel machines (not necessarily simultaneously) with a goal of minimizing the makespan. In 1969 Graham showed that processing jobs in decreasing order of size, assigning each to the currently-least-loaded machine, yields a 4/3-approximation for $P||C_{max}$. Our main result is a proof that the natural generalization of Graham's algorithm also yields a 4/3-approximation to the minimum-space advertisement scheduling problem. Previously, this algorithm was only known to give an approximation ratio of 2, and the best known approximation ratio for any algorithm for the minimum-space ad scheduling problem was 3/2. Our proof requires a number of new structural insights, which leads to a new lower bound for the problem and a non-trivial linear programming relaxation. We also provide a pseudo-polynomial approximation scheme for the problem (polynomial in the size of the problem and the number of machines).

1 Introduction

We study a scheduling problem whose application is the optimal placement of advertisement images within a shared space over time, typically on a web page. Roughly \$3 billion was spent on web advertising in the first half of 2002 alone [7], so improvements in algorithms for ad placement are of both economic and theoretical interest. In the model we consider here, we have a set $\mathcal{A} = [n] := \{1, \cdots, n\}$ of n ads (images of fixed width and varying heights) to schedule within a shared vertical space, typically in the margin of a web page. We must determine the subset of ads to display in each of T occurences of the page, or T time slots. Ad i has a height h_i and a display count $c_i \leq T$ which represents the number of time slots out of T in which the ad must appear. The goal is to assign each ad i to a set of c_i distinct time slots so as to minimize the maximum height of any occurrence of the page, as illustrated graphically in Figure 1. Mathematically, this means that we need to find $A_i \subseteq [n]$ for $i \in [n]$ such that (i) $|A_i| = c_i$ for every i and (ii) $\max_{t \in [T]} \sum_{i: t \in A_i} h_i$ is minimized. This problem was first posed in 1998 by Adler, Gibbons, and Matias [1]. Notice that we do not care about the vertical ordering of the ads within a single time slot.

When $c_i = 1$ for every i, the problem reduces to the classical NP-Hard scheduling problem $P||C_{max}$ with the time slots corresponding to machines, the ads corresponding to jobs and the height h_i corresponding to the processing time p_i. In fact, our ad scheduling problem is very similar to $P||C_{max}$ with *high-multiplicity encoding* of jobs with the same processing time, except that we require that these additional copies be scheduled on different machines. The high-multiplicity encoding of $P||C_{max}$, denoted by $P|M|C_{max}$ (see Clifford and Posner [2]) has been the focus of some study in the 90's, but there are still many open questions surrounding this problem, in particular whether or not the complexity of an optimal solution is of size polynomial in the input. See McCormick et al. [9] and Clifford and Posner [2] for further discussion.

* This work was supported by NSF contracts CCR-0098018 and ITR-0121495.

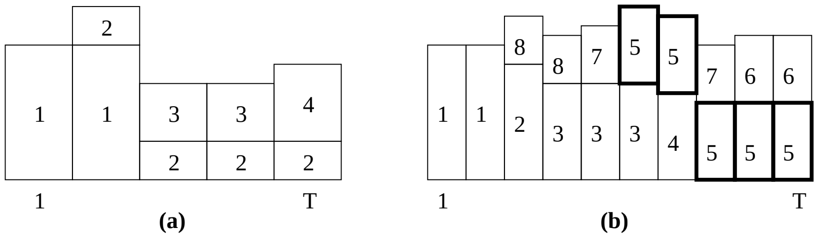

Fig. 1. Sample schedules. The vertical axis represents vertical space and each time slot corresponds to a column along the horizontal axis. (a) LSLF schedule for which Graham's proof fails, (b) Illustration of wrap-around behavior of LSLF when scheduling large ads.

For $P||C_{max}$, Graham [4] shows that processing the jobs in any order and greedily placing each job in sequence on the currently-least-loaded machine is a 2-approximation algorithm and that this is tight. He further shows that greedy processing of jobs with largest processing time first (LPT) is a 4/3-approximation algorithm and this is also tight. For the ad scheduling problem, Adler, Gibbons, and Matias [1] introduce the analogue of LPT, the *Largest Size Least Full (LSLF)* algorithm which processes ads in non-increasing order of height and greedily schedules ad i on the c_i currently-least-loaded time slots, and show that it is a 2-approximation algorithm, leaving open the question whether a better approximation factor could be proved. Subsequently, Dawande, Kumar, and Sriskandarajah [3] show that any list processing of ads in a greedy fashion is a 2-approximation algorithm, and that this is tight. Dawande et al. also show that rounding the optimum solution of a trivial linear programming relaxation of the problem leads to a 2-approximation algorithm and that a more sophisticated relaxation can be used to obtain a 3/2-approximation algorithm. Previous to this paper, these were the best known approximation algorithms for the ad scheduling problem. One of the main results of this paper is to show that LSLF is a 4/3-approximation algorithm, thereby matching Graham's bound for $P||C_{max}$ (this is tight, due Graham's analysis). Our proof is, however, much more elaborate than for the case of $P||C_{max}$.

Theorem 1. *LSLF is a 4/3-approximation algorithm for the ad scheduling problem.*

Regarding approximation schemes for $P||C_{max}$, for case when the number of machines is a constant Horowitz and Sahni [6] gave a $(1+\epsilon)$-approximation algorithm for any $\epsilon > 0$ (PTAS). This was improved by Hochbaum and Shmoys [5] to all instances of $P||C_{max}$. Since the running time of this algorithm depends polynomially on the number of machines, it is only a pseudopolynomial approximation scheme for $P|M|C_{max}$. We introduce a similar pseudopolynomial approximation scheme for the ad scheduling problem, whose running time is polynomial in n and T.

Discussion of Our Results. Graham's proof for the performance guarantee of 4/3 for LPT differentiates *large* jobs with processing times greater than $OPT/3$ (OPT denotes the makespan of an optimal schedule), from the remaining *small* jobs. He observes the following: (i) if the makespan of the LPT schedule is defined by a machine containing only large jobs, then the LPT schedule will in fact be optimal; (ii) on the other hand, if the makespan of the LPT schedule is defined by a machine whose final job j is small, then the makespan can be written as $L + p_j$, where L denotes the load of that machine prior to j's scheduling. The 4/3 bound then follows since $p_j \leq OPT/3$ and since $L \leq OPT$ because greedy scheduling ensures that every other machine must have a load of at least L.

We will show that (i) also holds for the ad scheduling problem: if a maximum-height slot in the LSLF schedule contains only large ads (having heights greater than $OPT/3$), then the LSLF schedule is optimal. However, in (ii) the assumption that $L \leq OPT$ no longer holds for the ad scheduling case, as shown in Figure 1(a).

Regarding (i), we will prove in Theorem 2 the following stronger result: if all ads are large, then LSLF not only minimizes the height of the largest time slot, but it also minimizes the sum of the heights of the largest k time slots for any k over all schedules that place at most 2 ads per time slot (and this is the case indeed for the optimum schedule). To the best of our knowledge, this structural result was not even known for $P||C_{max}$. We use this to devise a stronger lower bound based on the following intuition: consider a set P_k comprising k time slots into which LSLF places the greatest amount of large ad volume. By our structural result, the total large ad volume placed in P_k by LSLF is a lower bound on the large ad volume found in P_k in an optimal schedule. Further, we can lower bound the amount of small ad volume that must appear in P_k by noting that for each small ad i, we must schedule at least $\max(c_i - (T-k), 0)$ copies of i in P_k. Therefore, if we are able to locate such a set P_k for which (i) every small ad i is scheduled by LSLF in exactly $\max(c_i - (T-k), 0)$ slots in P_k, and (ii) the maximum and minimum slot heights in P_k differ by at most $OPT/3$, then the 4/3 approximation bound would follow. Unfortunately, there are problem instances in which no such set P_k satisfies these conditions. In these cases, we manage to prove that there exists a subset $\mathcal{A}'$ of our set $\mathcal{A}$ of ads and a value k such that (i) the values of LSLF on $\mathcal{A}$ and $\mathcal{A}'$ are equal, $LSLF(\mathcal{A}) = LSLF(\mathcal{A}')$, and (ii) the above property holds for $\mathcal{A}'$ and k.

We will also present a different 4/3-approximation algorithm based on rounding a new linear programming relaxation built on our structural results above. We will show an efficient combinatorial technique for solving this LP, and that its solution is exactly the closed-form expression of our aforementioned lower bound.

Finally, we describe a PTAS based on a combination of dynamic programming and LP rounding, which is similar to other approximation schemes for $P||C_{max}$. Its running time is polynomial in n and T, and therefore only pseudopolynomial. Our two 4/3-approximation algorithms, in contrast, can be implemented to run in time polynomial in n with no dependence on T; details are left for the full version. For that purpose, the output is condensed by listing every distinct assignment of ads to a time slot along with the number of time slots with this assignment.

The extended abstract is structured as follows. In the next section, we study the LSLF schedule and shows that it schedules the large ads extremely well. We also deduce our new lower bound. In section 3, we prove that LSLF produces a schedule within a factor of 4/3 of the optimum. In Section 4, we introduce our linear programming relaxation, show that its value is equal to our lower bound and derive rounding-based 4/3-approximation algorithms. Finally, in Section 5, we sketch a pseudo-polynomial approximation scheme for the problem.

2 Large Ads in the LSLF Schedule

We declare ad i to be *large* if $h_i > OPT/3$, and *small* if $h_i \leq OPT/3$. Observe that the total number of large ad copies in our problem instance cannot exceed $2T$ since any optimal solution must contain no more than two large ads per slot.

Lemma 1. *The LSLF algorithm places at most two large ads in every slot.*

Proof. Consider a division of the large ads into two sets: the set of *huge* ads containing ads i for which $h_i > 2OPT/3$, and the remaining set which we call the *medium* ads. LSLF first schedules the huge ads, one copy per slot. Let P denote the set of slots occupied by these ads, and Q denote the remaining slots. LSLF then schedules the medium ads one by one. We claim by induction that the following two invariants hold during this process: (i) all medium ads are placed solely within slots in Q, and (ii) at most two medium ads are placed in each slot in Q.

Suppose our two invariants hold up to the point where LSLF schedules a particular medium ad i. We must have $c_i \leq |Q|$, otherwise this ad would share a slot with some huge ad in every optimal solution, which clearly cannot happen. Therefore, in principle we have sufficient room to fit all copies of ad i within only the slots in Q. We now claim that there must be c_i slots in Q which currently contain at most one medium ad. If this were not the case, then the total number of medium ads would exceed $2|Q|$, and this would imply that in every optimal solution either (i) a huge and medium ad must share a slot, or (ii) there must be a slot containing more than two medium ads; both of these situations are clearly impossible. All slots in Q having at most one medium ad are currently shorter than slots in Q having two medium ads and also shorter than slots in P, so LSLF selects these slots for the placement of ad i. This maintains both of our invariants.

Not only does LSLF place at most two large ads in any slot, but it arranges these ads according to a very specific structure. LSLF schedules one row of large ads and then wraps around to schedule another row of large ads in reverse. The wrap-around point is a bit delicate though, as shown in Figure 1(b), because we must avoid placing two copies of the same ad in a common slot. One can argue that the LSLF schedule for the large ads can be constructed in the following simple recursive way. (i) If an ad exists with multiplicity T, then we schedule this ad in every slot and then any remaining large ad one copy per slot; (ii) if there are $2T$ large ad copies, we pair up a copy of one of the maximum-height large ads with a copy of one of the minimum-height large ads, decrease T, update the multiplicities and then recurse; (iii) if there are fewer than $2T$ large ad copies, we place a copy of a maximum-height large ad by itself and recurse.

We refer to the part of a schedule consisting of only large ads as the *base* of the schedule. We have just described a simple recursive way to construct the base of the LSLF schedule. For any schedule S, let $b_1(S) \geq b_2(S) \geq \cdots \geq b_T(S)$ denote the height occupied by the base, namely the large ads, in every time slot, ordered in a non-increasing fashion. Also, for $k = 1, \cdots, T$, define $B_k(S) = \sum_{i=1}^{k} b_i(S)$. We claim that LSLF schedules the large ads in a very balanced way, at least as well as in the optimal schedule. This is formalized in the following theorem.

Theorem 2. *Let LSLF denote the schedule produced by the LSLF algorithm, and let Σ be an optimum schedule. Then for all values of k, $B_k(LSLF) \leq B_k(\Sigma)$.*

Proof. The proof is by contradiction. Suppose we have a schedule S generated by $LSLF$, an optimal schedule Σ, and a value k for which the lemma fails. We will show how to transform the base of Σ into the base of S without increasing $B_k(\Sigma)$. Based on our recursive characterization of how LSLF schedules the large ads, we will make the bases of S and Σ agree in one slot — for example by making an exchange in Σ so a maximum-height large ad is paired with a minimum-height large ad. We will then recursively transform the remaining $T-1$ slots of the base of Σ into the remaining $T-1$ slots of the base of S. Our transformation must consider these three cases:

1. If a large ad i exists for which $c_i = T$, then LSLF will schedule this ad in every slot, and the remaining large ads one copy per slot. In Σ, we will also find ad i scheduled once in every slot, and every other large ad scheduled one copy per slot, since the optimal schedule cannot have more than two large ads per slot. Therefore, if a large ad of multiplicity T exists, then the base of Σ will be the same as the base of S. We can terminate our recursive transformation once we reach this case.
2. Suppose that $c_i < T$ for every large ad i and that there are exactly $2T$ large ad copies. If there exists a slot in Σ in which a maximum-height large ad is paired with a minimum-height large ad, then we can hold this slot fixed, decrement T and the display counts of the two ads in this slot, and continue our recursive transformation on the remaining slots. Otherwise, suppose we can locate distinct large ads i and j such that a maximum-height large ad is paired with a copy of i and a minimum-height large ad is paired with a copy of j. Swapping ad i and the

minimum-height large ad with each-other will change the pairing from (Max, i), (Min, j) to (Max, Min), (i, j), and will not decrease $B_k(\Sigma)$ for any k. We may then continue our recursive transformation. If none of the above cases apply, then every instance of a maximum-height or minimum-height large ad must be paired with the same large ad l. However, since $c_l < T$ there must be some slot in which two large ads i and j different from l are paired. Assume $h_i \leq h_j$. If $h_i < h_l$, then we make a swap to change the pairing (Max, l), (i, j) to (Max, i), (l, j). If $h_i \geq h_l$, we make a swap to change (Min, l), (i, j) into (Min, j), (i, l). This exchange also will not decrease $B_k(\Sigma)$ for any k, and will place us in the prior situation where one more swap suffices to pair up the mininimum-height and maximum-height large ads.

3. Suppose that $c_i < T$ for every large ad i and that there are fewer than $2T$ large ad copies. If there exists a slot in the base of Σ in which the only large ad is a maximum-height large ad, then we hold this slot fixed, decrement T and the display count of the ad in this slot, and continue our recursive transformation on the remaining slots. Otherwise, we find a slot containing a maximum-height large ad and swap away the remaining large ad in its slot. The details are essentially the same as in case 2.

One can easily prove the following corollaries to Theorem 2.

Corollary 1. *If all ads are large, LSLF produces an optimal schedule.*

Corollary 2. *Consider the schedule produced by LSLF. If a maximum-height slot in this schedule contains only large ads, then this schedule is optimal.*

2.1 A New Lower Bound on OPT

In order to argue approximation bounds we need to find suitable lower bounds on OPT. The maximum ad height, h_{max}, and the average slot height in any schedule, $\sum_i c_i h_i / T$, are both trivial lower bounds on OPT. These lower bounds are sufficient to show that LSLF is a 2-approximation algorithm [3]; however, in order to show that LSLF is a 4/3-approximation algorithm, we introduce a new, more powerful lower bound based on Theorem 2.

Theorem 3. *Let $\mathcal{A}_S$ denote the set of small ads in our problem instance. Then for every k, we have $L_k \leq OPT$, where*

$$L_k = \frac{1}{k}\left(B_k(LSLF) + \sum_{i \in \mathcal{A}_S} h_i \max(c_i - (T-k), 0)\right).$$

Proof. Consider k slots of maximum base height in the optimum schedule Σ. We know from Theorem 2 that the total base height $B_k(\Sigma)$ in these k slots is greater or equal to $B_k(LSLF)$. Now consider any small ad i. The smallest number of copies of i that Σ could possibly assign to these k slots is $\max(c_i - (T-k), 0)$ (since $T-k$ is an upper bound on the number of copies outside of these k slots). Thus the total height in Σ in these k slots is at least $B_k(LSLF) + \sum_{i \in \mathcal{A}_S} h_i \max(c_i - (T-k), 0)$ and the maximum height of any of these slots must be at least the average value L_k.

Observe that the two trivial lower bounds mentioned earlier are dominated by this bound: $h_{max} \leq L_1$ and $\sum_i c_i h_i / T = L_T$.

Although we do not know which ads are large and which are small, we can nevertheless compute a lower bound based on Theorem 3. If we know the index j of the smallest large ad, we can compute the bound $LB(j) = \max_k L_k$ given by the above Theorem. Since we do not know j, compute j^* to be the largest index such that $LB(j^*) \leq 3h_{j^*}$ or $j^* = 0$ if no such index exists, in which case there are no large ads. We claim that $LB(j^*)$ is a lower bound on OPT, and it is at least as good as the one if we knew which were the large ads. Indeed, it is a lower bound since either $h_{j^*} > OPT/3$ in which case j^* is a large ad and $LB(j^*)$ is a lower bound on OPT by Theorem 3, or $h(j^*) \leq OPT/3$ in which case $3h(j^*)$ is a lower bound on OPT and therefore so is $LB(j^*)$. Furthermore, j^*+1 is not the index of a large ad since otherwise $LB(j^*+1) \leq OPT < 3h_{j^*+1}$, contradicting the choice of j^*. Thus the unknown index j of the smallest large ad satisfies $j \leq j^*$. Finally, $LB(\cdot)$ can be shown to be non-decreasing (either directly from the formula for L_k given above, or using Theorem 5 in Section 4 and arguing that the way LSLF places job l in $LB(l)$ is feasible for the linear program corresponding to $LB(l-1)$). The fact that $j \leq j^*$ then implies that $LB(j^*)$ is at least as good a lower bound as $LB(j)$.

3 Analysis of the LSLF Algorithm

Throughout this section, we assume that the time slots are indexed in non-increasing order of base heights: $b_1(LSLF) \geq b_2(LSLF) \geq \cdots \geq b_T(LSLF)$. We first need several definitions. If P is a set of slots, we say that a small ad i is P-minimal if exactly $\max(c_i - (T - |P|), 0)$ copies of i are scheduled in P by LSLF. Thus, in the above argument we would like to have that all small ads are P-minimal, where $P = [k]$ is a prefix of the slots. For a time slot t, let $H_i(t)$ denote the height of t immediately after LSLF schedules ad i. For a set of slots P we then define $Min_i(P) = \min\{H_i(t) : t \in P\}$ and $Max_i(P) = \max\{H_i(t) : t \in P\}$. Finally, we define the *range* of a set of slots as $Range_i(P) = Max_i(P) - Min_i(P)$. For notational simplicity, we will omit the subscript on these quantities when we speak of the final schedule produced by LSLF, so for example $Min(P) = Min_n(P)$.

As alluded to in the introduction, if we could find a value k such that (i) $Range([k]) \leq OPT/3$ and (ii) every small ad is $[k]$-minimal then it would be easy to show that LSLF produces a schedule of value at most $L_k + OPT/3 \leq \frac{4}{3}OPT$; this is formalized in Lemma 4 below. Unfortunately, such a value k does not always exist, but we will show that we can find a subset of the ads for which the value of LSLF does not change and the above property is satisfied.

In order to guarantee that $Range(P) \leq OPT/3$ for certain sets P, we use the following Lemma and Corollary.

Lemma 2. *Let t and u be slots satisfying $t < u$. If i is a small ad or the last large ad processed, then $H_i(u) \leq H_i(t) + OPT/3$.*

Proof. Since $t < u$, we know that $b_t(LSLF) \geq b_u(LSLF)$, so u will be no taller than t just before scheduling all the small ads. As LSLF schedules small ads, as long as u remains no taller than t the lemma is certainly satisfied. So let us assume that at some point in time we schedule a small ad i in u but not in t, resulting in u becoming taller than t. However, at this point, u will be taller by at most $h_i \leq OPT/3$, so we have $Range_i(\{t,u\}) \leq OPT/3$. We can now observe that $H_i(u) - H_i(t)$ cannot increase as we increment i as long as $H_i(t)$ does not overcome $H_i(u)$. This shows that $Range(\{t,u\}) \leq OPT/3$ at termination.

Corollary 3. *Let k be a maximum-height slot in the final LSLF schedule. If a small ad was scheduled in k, then $Range([k]) \leq OPT/3$.*

We need one more concept. As the algorithm progresses, we will differentiate between *heavy* and *light* time slots with respect to a prefix $P = [k]$ of time slots. We assume that LSLF has already scheduled the large ads; so from now on, we focus on small ads only. At the beginning (so just before scheduling any small ad), we designate all time slots to be heavy; more formally, we say that all slots are $(P, 0)$-heavy. If we now consider the point in time right after LSLF schedules some (small) ad i, we say that a time slot is t is (P, i)-light if:

– Rule I: Slot t is $(P, i-1)$-light.
– Rule II: Ad i is small, not P-minimal, and is scheduled in some slot u for which $H_i(u) \geq H_i(t)$.
– Rule III: Ad i is small, P-minimal, and is scheduled in some $(P, i-1)$-light slot u for which $H_i(u) \geq H_i(t)$.

If none of these three conditions apply, then we say slot t is (P, i)-heavy. Let us briefly build some intuition about this definition. As soon as a slot becomes light, it will remain light forever. If a small ad i is not P-minimal, this immediately forces all slots receiving a copy of i to become light. This means that, at the end, all slots are P-heavy if and only if all small ads are P-minimal. Additionally, a slot becomes light any time we notice that another light slot matches or exceeds it in height, so the heavy slots always dominate the light slots in height. This is formalized by the following lemma.

Lemma 3. *If slot t is (P, i)-heavy and slot u is (P, i)-light, then $H_i(t) > H_i(u)$.*

Proof. Suppose the lemma fails for two slots t and u. Consider the smallest i for which t is (P, i)-heavy, u is (P, i)-light, and $H_i(t) \leq H_i(u)$. We know i is a small ad and that t is $(P, i-1)$-heavy by rule I. Consider the following cases:

1. **Slot u is $(P, i-1)$-light, and i is not P-minimal.** Since we picked i to be minimal, we know $H_{i-1}(t) > H_{i-1}(u)$, so i must be scheduled in u but not in t. Rule II thus implies that t must be (P, i)-light.
2. **Slot u is $(P, i-1)$-light, and i is P-minimal.** Again, since we picked i to be minimal, $H_{i-1}(t) > H_{i-1}(u)$ and i must be scheduled in u but not in t. Rule III thus implies that t must be (P, i)-light.
3. **Slot u is $(P, i-1)$-heavy, and i is not P-minimal.** By rule II, we know i is scheduled in some slot v for which $H_i(v) \geq H_i(u)$. However, since $H_i(v) \geq H_i(u) \geq H_i(t)$, rule II implies that t must be (P, i)-light.
4. **Slot u is $(P, i-1)$-heavy, and i is P-minimal.** By rule III, we know i is scheduled in some $(P, i-1)$-light slot v for which $H_i(v) \geq H_i(u)$. However, since $H_i(v) \geq H_i(u) \geq H_i(t)$, rule III again implies that t must be (P, i)-light.

In all cases, we conclude that t is (P, i)-light, a contradiction.

The following lemma formalizes our informal discussion earlier in this section of a case for which the 4/3 bound follows easily. As we said earlier, such a prefix of slots does not always exist.

Lemma 4. *If there exists a prefix $P = [k]$ of slots for which $Range(P) \leq OPT/3$ and for which all slots are (P, n)-heavy, then $Max([T]) \leq 4/3OPT$.*

Proof. Consider the lower bound L_k from Theorem 3. We argue that L_k will be the average slot height within P, since every small ad i must be P-minimal. If any small ad weren't P-minimal, all slots receiving it would have become light. Since the minimum slot height in P is a lower bound on the average slot height in P, we have $Min(P) \leq L_k$, and since $L_k \leq OPT$ by Theorem 3, $Min(P) \leq OPT$. Finally, since $Range(P) \leq OPT/3$, we have $Max(P) \leq Min(P) + OPT/3 \leq 4/3OPT$.

We are now equipped to give the main result of this paper, a proof of Theorem 1.

Proof. We first describe the core argument at a high level, postponing discussion of a few key technical lemmas. Consider the schedule produced by LSLF on some instance I. Let P be the largest possible prefix of slots one can form such that $Range(P) \leq OPT/3$. By Corollary 3, we know that P can at least be made large enough to capture every maximum-height slot. Consider now the following three cases.

1. **All slots are (P,n)-heavy.** In this case, Lemma 4 says that the makespan of the LSLF schedule is at most $4/3 OPT$.
2. **All slots are (P,n)-light.** This case is impossible. If all slots are P-light, then Lemma 5 below applied to the last small ad and the slot achieving the maximum height implies that we can extend P to a larger prefix P' for which $Range(P') \leq OPT/3$, thereby contradicting the maximality of P.
3. **Some slots are (P,n)-heavy and some are (P,n)-light.** In this case, we reduce our problem instance I to a strictly smaller instance I' by deleting a carefully chosen subset of the ads. Since I' contains fewer ads, we have $OPT(I') \leq OPT(I)$, and Lemma 6 below shows how to construct I' such that $LSLF(I') = LSLF(I)$. We now claim by induction on the size of our instance that $LSLF(I') \leq 4/3 OPT(I')$, so $LSLF(I) = LSLF(I') \leq 4/3 OPT(I') \leq 4/3 OPT(I)$.

This completes the proof. All that remains is to argue lemmas 5 and 6.

Lemma 5. *If slot t is (P,i)-light where $P = [k]$ then $H_i(t) - H_i(k+1) \leq OPT/3$.*

Proof. We proceed by induction on i and consider 4 different cases.

1. **Ad i is scheduled on t but not on $k+1$.** In this case, we have

$$H_i(t) = H_{i-1}(t) + h_i \leq H_{i-1}(k+1) + h_i \leq H_i(k+1) + h_i \leq H_i(k+1) + OPT/3.$$

2. **We are not in case 1, and t is $(P,i-1)$-light.** We know that $H_{i-1}(t) - H_{i-1}(k+1) \leq OPT/3$ by induction. Furthermore, since we are not in case 1, we have that $(H_i(t) - H_{i-1}(t)) - (H_i(k+1) - H_{i-1}(k+1)) \leq 0$, which summed up with the previous inequality gives the statement of the lemma.
3. **Slot t is $(P,i-1)$-heavy and i is P-minimal.** Since t is (P,i)-light, rule III must have applied. This means that i is scheduled on a $(P,i-1)$-light slot u with $H_i(u) \geq H_i(t)$. Since u must fall into either case 1 or 2 above, we have $H_i(u) - H_i(k+1) \leq OPT/3$. Therefore, $H_i(t) \leq H_i(u) \leq H_i(k+1) + OPT/3$.
4. **Slot t is $(P,i-1)$-heavy and i is not P-minimal.** So, rule II must have applied and i is scheduled on a slot u with $H_i(u) \geq H_i(t)$. Since i is not P-minimal, there exists $v \geq k+1$ such that i is not scheduled on v. We further consider two cases.
   (a) If v can be chosen to be $k+1$ then

$$H_i(t) \leq H_i(u) = H_{i-1}(u) + h_i \leq H_{i-1}(k+1) + h_i \leq H_i(k+1) + OPT/3.$$

   (b) If not, then we can assume that i is scheduled on $k+1$ and then we have

$$\begin{aligned} H_i(t) &\leq H_i(u) = H_{i-1}(u) + h_i \leq H_{i-1}(v) + h_i \\ &\leq H_{i-1}(k+1) + OPT/3 + h_i = H_i(k+1) + OPT/3, \end{aligned}$$

   where the last *inequality* follows from Lemma 2 since $v > k+1$.

Lemma 6. *Fix an instance I and a prefix P of slots. Suppose that both (P,n)-heavy and (P,n)-light slots exist. Create a new problem instance I' by deleting all small ads except those ads i having at least one copy scheduled in some (P,i)-heavy slot. Then (i) I' will be a strictly smaller instance than I, (ii) $OPT(I') \leq OPT(I)$, and (iii) $LSLF(I') = LSLF(I)$.*

Proof. We argue that (i) follows from the existence of (P,n)-light slots. In order for (P,n)-light slots to exist, there must be some non P-minimal small ad i, and every slot receiving a copy of i will be (P,i)-light by rule II. Thus i will be deleted when forming I'. Point (ii) is also straightforward, as deletion of ads can never cause OPT to increase. We therefore focus our attention on (iii).

Consider applying LSLF in parallel to simultaneously construct schedules for I and I'; we will compare corresponding slots in the two schedules as we do so. At any point in time after scheduling all copies of ad i, let $\mathcal{H}(i)$ denote the set of (P, i)-heavy slots (with respect to the schedule for I) and let $\mathcal{L}(i)$ denote the set of (P, i)-light slots (also with respect to the schedule for I). We inductively argue the following: after scheduling any ad i,

1. $H_i^I(t) = H_i^{I'}(t)$ for all $t \in \mathcal{H}(i)$, and
2. $H_i^I(t) \geq H_i^{I'}(t)$ for all $t \in \mathcal{L}(i)$.

The superscripts I and I' above refer to the schedule in which we are measuring the height of a slot. Otherwise stated, the heavy slots in the schedules I and I' will always agree in their heights, while the light slots in I's schedule will always upper bound their corresponding slots in the schedule for I'. Since every slot t maximizing $H_n(t)$ (in either schedule) will belong to $\mathcal{H}(n)$, this will imply $LSLF(I) = LSLF(I')$.

The large ads all belong to both I and I', and will be identically-scheduled for both instances. Consider therefore the insertion of an arbitrary small ad i, assuming that our inductive hypothesis holds for $i - 1$.

- $i \in I'$. Consider the schedule for instance I. Since i was not deleted, it is scheduled in some slot in $\mathcal{H}(i)$, and therefore also in some slot in $\mathcal{H}(i-1)$. By lemma 3, i is scheduled in every slot in $\mathcal{L}(i-1)$. The inductive hypothesis and lemma 3 together imply that $H_{i-1}^{I'}(t) < H_{i-1}^{I'}(u)$ for every $t \in \mathcal{L}(i-1)$ and $u \in \mathcal{H}(i-1)$; hence, i will also be scheduled in every slot in $\mathcal{L}(i-1)$ in the schedule for I'. This ensures the invariant is maintained for all $t \in \mathcal{L}(i-1)$. Since the heights of slots in $\mathcal{H}(i-1)$ agree between the two schedules at time $i-1$, ad i will be scheduled in analogous slots in $\mathcal{H}(i-1)$ among both schedules. Some of these slots will be in $\mathcal{H}(i)$; the rest will move to $\mathcal{L}(i)$. In either case, since the heights of corresponding slots agree, the invariant will also be maintained for these slots.
- $i \notin I'$. In this case, i is not scheduled in any slot in $\mathcal{H}(i)$, so it can only appear in I's schedule in slots in $\mathcal{L}(i-1)$ and in $\mathcal{H}(i-1) \backslash \mathcal{H}(i)$. By the invariant, heights of slots in these two sets in I's schedule are already upper bounds on their corresponding slots in the schedule for I', and we can only be strengthening this upper bound. Therefore the invariant is maintained.

4 A New Linear Programming Relaxation

Theorem 2 allows us to give a new linear programming relaxation for the ad scheduling problem. By rounding the solution to this linear program we obtain another 4/3-approximation algorithm. The linear program optimally (and fractionally) assigns the *small* ads on top of the base obtained by LSLF so as to minimize the tallest slot. Since we do not know OPT, we do not know which ads are large and which are small, but if the ads are sorted in decreasing order of height, the large ads must comprise some prefix of this list, so we must run our rounding algorithm on every prefix and take the best result. Henceforth, we can therefore assume the large ads are known.

The linear program is the following, where $\mathcal{A}_S$ denotes the set of small ads.

$$
\begin{aligned}
LP = \text{Minimize } & z \\
\text{subject to: } & \\
& \sum_{t \in [T]} x_{it} = c_i && i \in \mathcal{A}_S \\
& \sum_{i \in [n]} x_{it} h_i \leq z - b_t(LSLF) && t \in [T] \qquad (1) \\
& 0 \leq x_{it} \leq 1 && i \in [n], t \in [T].
\end{aligned}
$$

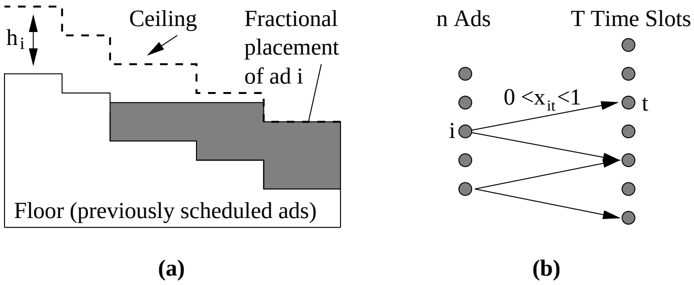

Fig. 2. Combinatorially solving the LP. (a) "Fluid" interpretation of fractional ad placement, (b) Bipartite graph of fractional assignments used for rounding.

It is not straightforward that this linear program is a lower bound on the optimum, since the optimum schedule might schedule the base differently than LSLF. We know of two ways of arguing that LP is a lower bound on OPT. The first proof is based on (contra)-polymatroids and is given below for completeness; the second follows from Theorems 5 and 6 where a simple combinatorial algorithm is shown to solve the LP optimally and with optimal value $\max_k L_k$, the lower bound given in Theorem 3.

Theorem 4. *$LP \leq OPT$.*

Proof. Let C denote the set of all vectors $l \in R^T$ such that the linear program with the right-hand-side of equation (1) replaced by l_t (from $z - b_t(LSLF)$) is feasible. By definition of Σ, we have that $s \in C$ where $s_t = OPT - b_t(\Sigma)$. By symmetry of the time slots and the fact that C is a convex set, this implies that

$$C \supseteq P := conv\left\{x : x_i = s_{\sigma(i)} \text{ for some permutation } \sigma \text{ of } [T]\right\} + R^T_+.$$

This latter polyhedral set P is a contra-polymatroid (see [10]) and can be completely described by inequalities in the following way:

$$P = \{x \in R^T : x(S) \geq g(|S|) \text{ for all } S \subseteq [T]\},$$

where $x(S) = \sum_{t \in S} x_t$ and $g(k)$ is the sum of the k smallest s_t, i.e. $g(k) = kOPT - B_k(\Sigma)$ using the notation introduced before Theorem 2. We claim that the right-hand-side of (1) for $z = OPT$, i.e. $OPT - b_t(LSLF)$, is in P, hence in C, and therefore $LP \leq OPT$. Indeed, for any set S,

$$\sum_{t \in S}(OPT - b_t(LSLF)) \geq |S|OPT - B_{|S|}(LSLF) \geq |S|OPT - B_{|S|}(\Sigma) = g(|S|)$$

by Theorem 2.

In the next theorem, we show that LP is equal to the lower bound given in Theorem 3. The proof of that theorem actually shows that, for any k, $LP \geq L_k$. This implies that $LP \geq \max_{k \in [T]} L_k$. The converse also holds:

Theorem 5. *$LP = \max_{k \in [T]} L_k$.*

In order to prove this theorem, we give a simple combinatorial algorithm that solves the LP and show that it has the right value. The algorithm first initializes the height H_t of each slot t

to be $b_t(LSLF)$ and processes the small ads in any order. The process of scheduling each small ad i can be thought of as filling the top of the schedule with a fluid, as shown in Figure 2(a). Ad i is fractionally "poured" onto the slot with minimum height until this slot catches up to the second-shortest slot and both are filled together uniformly, and so on. However, we must prevent the height of this fluid in each slot from exceeding h_i; this is done by imposing a "ceiling" at height h_i over the top of each slot, at which the fluid stops. When a total of c_i units of ad i have been fractionally filled in, we update $H_1 \dots H_T$ and the ceilings, and continue filling in the next ad.

Theorem 6. *The "fluid" algorithm generates an optimal solution to LP.*

Proof. In order to show that this algorithm optimally solves LP, consider the fractional solution it obtains at the end of its execution and the heights H_t of the time slots. By construction, $z = H_1 \geq H_2 \geq \cdots \geq H_T$. Let k be the maximum slot such that $H_k = H_1 = z$. During the execution of the algorithm, if an ad is ever fractionally assigned to any two adjacent slots t and $t+1$ (that is, if strictly less than its full height is assigned to both slots) then the heights of t and $t+1$ will remain equal for the remainder of the algorithm. In other words, since $H_k > H_{k+1}$, no ad was fractionally assigned to k and $k+1$ simultaneously. Furthermore since the algorithm assigns at least as much of an ad to time slot $t+1$ compared to t, no ad is assigned to any time slot in $[k]$ unless it is assigned fully to each of the time slots $k+1, \cdots, T$. This means that every small ad is $[k]$-minimal and we thus have that $H_1 + H_2 + \cdots + H_k = kL_k$. The fact that $H_1 = H_2 = \cdots = H_k$ by our choice of k implies that $z = L_k$. This simultaneously proves the correctness of the algorithm for solving the LP and of Theorem 5.

Approximation algorithms with performance guarantee of 4/3 can be obtained by solving this linear program and rounding the solution using classical rounding schemes such as the ones by Lenstra, Shmoys and Tardos [8] or by Shmoys and Tardos [11]. For example, consider in Figure 2(b) the bipartite graph corresponding to the fractional assignments produced by LP. That is, we have an edge from ad i to slot t if $0 < x_{it} < 1$. We assume without loss of generality that the edges in this subgraph form a forest, for if there were an alternating cycle among these edges we could "augment the flow" appropriately around such a cycle, maintaining feasibility and optimality of our solution, until the x_{it} value on one of its edges reached either 0 or 1, thereby breaking the cycle. In this graph, the outdegree of every ad i must be at least 2, since we have $\sum_t x_{it} = c_i$ and c_i is an integer. Therefore, we must be able to find an alternating path with endpoints at two different slots t and t', such that both t and t' have indegree of 1. We round our solution by augmenting flow on such alternating paths until all x_{it} values eventually become integral. During the process we only increase the flow entering a slot t if there is a single fractional edge (i, t) directed into t, so the total increase in height of any such slot will be at most $h_i \leq OPT/3$. Therefore, rounding increases the makespan of our solution by at most the maximum height of a fractionally-scheduled ad, which in this case is at most $OPT/3$. As described, our rounding approach takes time polynomial in n and T, but it is straightforward to eliminate the dependence on T by appropriate grouping of time slots. Further details are left for the full version.

5 A Pseudopolynomial $(1+\epsilon)$-Approximation Scheme

We describe briefly a $(1+\epsilon)$-approximation scheme whose running time is polynomial in n and T. The algorithm is similar to that of Hochbaum and Shmoys [5] for $P||C_{max}$. Let $\alpha = \epsilon/2$. For this section, we designate ads with heights larger than αOPT large, and the remaining ads small. As in the LP rounding case, we do not know which ads are large, so we must try our algorithm on every prefix of the sorted ads and take the best result. Let us therefore assume the large ads are known.

Our approximation scheme schedules the large ads via dynamic programming. We first run the greedy $LSLF$ algorithm to obtain approximate bounds on OPT, so $3LSLF/4 \leq OPT \leq LSLF$. The heights of the large ads are first rounded up to the nearest multiple of $3\alpha^2 LSLF/4$. Since large ads have heights at least αOPT, this inflates the height of each large ad (and hence also the optimal makespan) by a factor of at most $1+\alpha$. After rounding, we have at most $\lceil \frac{LSLF}{3\alpha^2 LSLF/4} \rceil = O(1)$ distinct large ad sizes. We can encode each base shape as a vector $(n_0, n_1, \cdots, n_{\lceil 4/3\alpha^2 \rceil})$ where n_i gives the number of slots having base height $3i\alpha^2 LSLF/4$, and $\sum n_i = T$. The number of distinct base shapes will therefore be at most $T^{\lceil 4/3\alpha^2 \rceil} = T^{O(1)}$, a polynomial in T. All achievable base shapes can be enumerated via dynamic programming in time polynomial in n and T. We omit further details. We henceforth assume that the base of a $(1+\alpha)$-approximate solution is known.

Fixing such a base, we solve LP to fractionally schedule the small ads. The optimal value of LP gives a lower bound on $(1+\alpha)OPT$, and rounding of the small ads results in an increase of at most αOPT. Thus, our final schedule has makespan at most $(1+\epsilon)OPT$.

Acknowledgements

We wish to thank the reviewers of this paper for their insightful comments.

References

1. M. Adler, P.B. Gibbons, and Y. Matias (1998). "Scheduling Space-Sharing for Internet Advertising". *Journal of Scheduling*. To appear.
2. J. Clifford, M. Posner (2001). "Parallel Machine Scheduling with High Multiplicity". *Mathematical Programming Ser. A* 89, 359–383.
3. M. Dawande, S. Kumar, and C. Sriskandarajah (2001). "Algorithms for Scheduling Advertisements on a Web Page: New and Improved Performance Bounds" *Journal of Scheduling*. To appear.
4. R. Graham (1969). "Bounds on Multiprocessing Timing Anomalies". *SIAM Journal on Applied Mathematics* 17, 416–429.
5. D. Hochbaum and D. Shmoys (1987). "Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results". *Journal of the ACM* 34, 144–162.
6. E. Horowitz and S. Sahni (1976). "Exact and Approximate Algorithms for Scheduling Nonidentical Processors". *Journal of the ACM* 23, 317–327.
7. Interactive Advertising Bureau (www.iab.net). IAB Internet Advertising Revenue Report.
8. J.K. Lenstra, D.B. Shmoys, and É. Tardos (1990). "Approximation algorithms for scheduling unrelated parallel machines". *Mathematical Programming* 46, 259–271.
9. S. T. McCormick, S. Smallwood, and F. Spieksma (2001). "A Polynomial Algorithm for Multiprocessor Scheduling with Two Job Lengths". *Mathematics of Operations Research* 26(1), 31–49.
10. A. Schrijver (2003). "Combinatorial Optimization - Polyhedra and Efficiency". Springer-Verlag.
11. D.B. Shmoys and É. Tardos (1993). "An Approximation Algorithm for the Generalized Assignment Problem". *Mathematical Programming* 62, 461–474.