OPT VERSUS *LOAD* IN DYNAMIC STORAGE ALLOCATION*

ADAM L. BUCHSBAUM†, HOWARD KARLOFF†, CLAIRE KENYON‡, NICK REINGOLD†, AND MIKKEL THORUP†

Abstract. DYNAMIC STORAGE ALLOCATION is the problem of packing given axis-aligned rectangles into a horizontal strip of minimum height by sliding the rectangles vertically but not horizontally. Where $L = LOAD$ is the maximum sum of heights of rectangles that intersect any vertical line and OPT is the minimum height of the enclosing strip, it is obvious that $OPT \geq LOAD$; previous work showed that $OPT \leq 3 \cdot LOAD$. We continue the study of the relationship between OPT and $LOAD$, proving that $OPT = L + O((h_{\max}/L)^{1/7})L$, where $h_{\max}$ is the maximum job height. Conversely, we prove that for any $\epsilon > 0$, there exists a $c > 0$ such that for all sufficiently large integers $h_{\max}$, there is a DYNAMIC STORAGE ALLOCATION instance with maximum job height $h_{\max}$, maximum load at most L, and $OPT \geq L + c(h_{\max}/L)^{1/2+\epsilon}L$, for infinitely many integers L. En route, we construct several new polynomial-time approximation algorithms for DYNAMIC STORAGE ALLOCATION, including a $(2+\epsilon)$-approximation algorithm for the general case and polynomial-time approximation schemes for several natural special cases.

Key words. Approximation algorithms, dynamic storage allocation, polynomial-time approximation schemes.

AMS subject classifications. 68Q17, 68Q25, 68W25, 68W40

1. Introduction.

We study a simple rectangle-packing problem: Given a set of n isothetic (i.e., axis-parallel) open rectangles in the x-y plane, the ith extending from x-coordinate x_i to x-coordinate y_i on the real line and having height h_i, slide each rectangle up or down *but not sideways* so that (1) each rectangle is a subset of the positive quadrant, (2) the regions of the plane they occupy are pairwise disjoint, and (3) the supremum of the y-coordinates used is minimized. Resembling off-line Tetris in which the rectangles slide vertically but not horizontally, this problem is formally known as DYNAMIC STORAGE ALLOCATION for the following reason. View a rectangle starting at x-coordinate x_i, ending at x-coordinate y_i, and of height h_i as a request for h_i contiguous bytes of storage starting at time x_i and ending at time y_i. Then the problem of finding a rectangle placement that minimizes the supremum of the y-coordinates used is exactly that of minimizing the number of contiguous bytes needed to satisfy all memory requests.

Formally, an instance of DYNAMIC STORAGE ALLOCATION is a set of some number n of *jobs*, each job i being an open interval $I_i = (x_i, y_i)$ (for rationals $x_i < y_i$), which we assume without loss of generality is a subset of $(0,1)$, and a positive rational *height* h_i. We say job i is *live at x-coordinate t* (or simply *live at t*) if $t \in I_i$. A feasible solution is an assignment of a nonnegative real $s(i)$ to each job i such that if we define $S(i)$ to be the open real interval $(s(i), s(i)+h_i)$ (the region of memory assigned to job i during time period (x_i, y_i)), then for all $t \in (0,1)$, the jobs i that are live at t have pairwise disjoint $S(i)$'s. The goal is to minimize the *makespan*: $\max_i\{s(i)+h_i\}$. By scaling, we will assume that each h_i is integral, in which case it is easy to see that without loss of generality $s(i)$ is integral too.

DYNAMIC STORAGE ALLOCATION was proven NP-Complete in 1976 by Stock-

* An extended abstract appears in *Proc. 35th ACM Symp. on Theory of Computing*, 2003.

†AT&T Labs, Shannon Laboratory, 180 Park Ave., Florham Park, NJ 07932, USA; email: {alb,howard,reingold,mthorup}@research.att.com.

‡Laboratoire d'Informatique, Ecole Polytechnique, 91128 Palaiseau Cedex, France; email: kenyon@lix.polytechnique.fr.

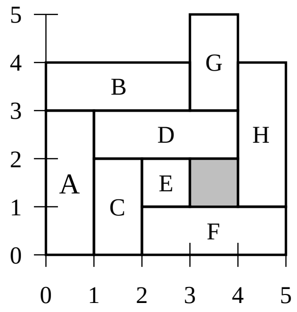

FIG. 1.1. *An optimal packing for the jobs (in the form (x_i, y_i, h_i)) $A = (0,1,3)$, $B = (0,3,1)$, $C = (1,2,2)$, $D = (1,4,1)$, $E = (2,3,1)$, $F = (2,5,1)$, $G = (3,4,2)$, and $H = (4,5,3)$. The shaded region is a gap. In this example, $LOAD = 4$, but $OPT = 5$.*

meyer. (See problem SR2 in Garey and Johnson [3].) The first polynomial-time, constant-factor approximation algorithm was given by Kierstead in 1988 [6], using a reduction from DYNAMIC STORAGE ALLOCATION to on-line coloring of interval graphs discovered by Woodall in 1973 [8] and independently by Chrobak and Slusarek in 1988 [1]. Kierstead's algorithm is an 80-approximation algorithm. Kierstead [7] later exhibited a 6-approximation algorithm for DYNAMIC STORAGE ALLOCATION. The next improvements were 5- and 3-approximation algorithms by Gergov [4, 5]. Neither of Gergov's two algorithms uses the Woodall-Chrobak-Slusarek reduction.

Define OPT to be the optimal makespan for a given instance. The *load $L(t)$ at (x-coordinate) t* is the sum of h_i over all jobs i that are live at t. The *maximum load* $LOAD$ (also L) is the maximum over all t of the load at t. $LOAD$ is a trivial lower bound on OPT. When all heights are equal, $OPT = LOAD$, and an optimal solution can be found via interval graph coloring. In general, however, OPT need not equal $LOAD$, as exemplified by Figure 1.1.

All four of the algorithms mentioned above actually find packings of makespan at most c times $LOAD$, not just c times OPT, for the respective c (80, 6, 5 or 3). We further study the relationship between OPT and $LOAD$ in DYNAMIC STORAGE ALLOCATION, providing new upper and lower bounds on the gap between them; all of our upper bounds are expressed algorithmically. Recall $L = LOAD$; denote the maximum job height by $h_{\max}$ and the minimum job height by $h_{\min}$. Our main results are as follows.

1. We devise an algorithm that yields makespan $(1 + O((h_{\max}/L)^{1/7}))L$. This bound is $L + o(L)$ when $h_{\max} = o(L)$. When $h_{\max}$ is bounded by a constant, we improve the makespan bound to $(1+O(((\lg^2 L)/L)^{1/4}))L$. Note that the case $h_{\max} = o(L)$ does not subsume that of $h_{\max}$'s being a constant, because, e.g., L might itself be bounded by a constant. No result of the form $OPT \le L + o(L)$ is possible in the general case, since one can "scale up" the heights in the example of Figure 1.1 to get $L = 4k$ and $OPT = 5k$ for any positive integer k.

2. For any $\epsilon > 0$, there exists a $c > 0$ such that for all sufficiently large constants $h_{\max}$, there is a DYNAMIC STORAGE ALLOCATION instance with maximum job height $h_{\max}$, maximum load at most L, and $OPT \ge (1 + c(h_{\max}/L)^{1/2+\epsilon})L$, for infinitely many integers L. This is the first non-trivial lower bound to include the case when $h_{\max}$ is bounded by a constant. This lower bound shows that our upper bounds in (1) are optimal up to the exact powers of $h_{\max}/L$. In particular, the "1/7" in the general upper bound (and the "1/4" in the bounded-$h_{\max}$ upper bound) cannot be replaced

by any real greater than 1/2.

We also contribute the following corollaries to our main results.

1. For all $\epsilon > 0$, we give polynomial-time, $(2+\epsilon)$-approximation algorithms.
2. We give polynomial-time approximation schemes for the following cases:
   (a) $h_{\max} = o(L)$;
   (b) $h_{\max}$ is bounded by a constant.

Finally, using straightforward dynamic programming, we give a PTAS when $h_{\min} = \Omega(L \lg\lg n/\lg n)$ and a polynomial-time, exact solution when $L = O(\lg n/\lg\lg n)$.

We begin in §2 by introducing our main tool: boxing jobs into larger rectangles so that the total wasted space in the rectangles remains small. Along with some results for simple cases (small load and large jobs) described in §3, we use boxing to devise our algorithms for bounded-height jobs (§4) and finally for the remaining cases (§5). We present the lower bound in §6.

2. Boxing Jobs. Let Y be any set of jobs and t be any x-coordinate. $L_Y(t)$ denotes the load of the jobs in Y that are live at t.

To *box* a set Y of jobs means to place the jobs into a box b of minimum x-coordinate $x_b = \min\{x_j : j \in Y\}$, maximum x-coordinate $y_b = \max\{y_j : j \in Y\}$, and height $h_b \geq \sum_{j \in Y} h_j$. A *boxing* of a set Y into a set B of boxes is a partition of Y into at most $|B|$ subsets, each of which is then boxed into a distinct box $b \in B$. The boxes can be viewed as jobs in a modified instance. Then L_B (resp., $L_B(t)$) is well defined to mean the *load of the boxes* (resp., *at* t). In particular, any unused space in a box still counts toward the load contributed by that box. All of the boxing procedures that follow run in polynomial time.

2.1. Boxing for a Fixed Time.

LEMMA 2.1. *Given a set Y of unit-height jobs, all live at some fixed x-coordinate t, an integer* box-height parameter H, *and a sufficiently small positive ϵ, there exist a subset Y' of Y, $|Y - Y'| \leq 2H\lceil 1/\epsilon^2\rceil$, a set B of boxes, each of height H, and a boxing of Y' into B such that at any x-coordinate u,*

$$L_B(u) \leq L_{Y'}(u) + 4\epsilon L_Y(u).$$

Proof. It is convenient to view a job starting at x and ending at y as a point (x, y) in the plane, as depicted in Figure 2.1(a)–(b). Now partition the jobs of Y into *strips*, as exemplified by Figure 2.1(c). The first two strips are defined as follows.

1. Create a vertical strip consisting of the $H\lceil 1/\epsilon^2\rceil$ jobs with the earliest starting x-coordinates (or fewer if there are not enough jobs).
2. If any jobs remain, create a horizontal strip consisting of the $H\lceil 1/\epsilon^2\rceil$ jobs that remain with the latest ending x-coordinates (or fewer if not enough jobs remain).

Define Y' to be the set of all jobs in neither the first vertical nor the first horizontal strip. Obviously $|Y - Y'| \leq 2H\lceil 1/\epsilon^2\rceil$. Now partition the jobs of Y' into strips by repeating the following as long as jobs remain.

1. Create a vertical strip consisting of the $H\lceil 1/\epsilon\rceil$ jobs that remain with the earliest starting x-coordinates (or fewer if not enough jobs remain).
2. If any jobs remain, create a horizontal strip consisting of the $H\lceil 1/\epsilon\rceil$ jobs that remain with the latest ending x-coordinates (or fewer if not enough jobs remain).

Now for every vertical strip of Y', take the jobs in order of decreasing ending x-coordinate in groups of size H (the last group of the last strip possibly smaller), and box them. Similarly, for every horizontal strip of Y', take the jobs in order of

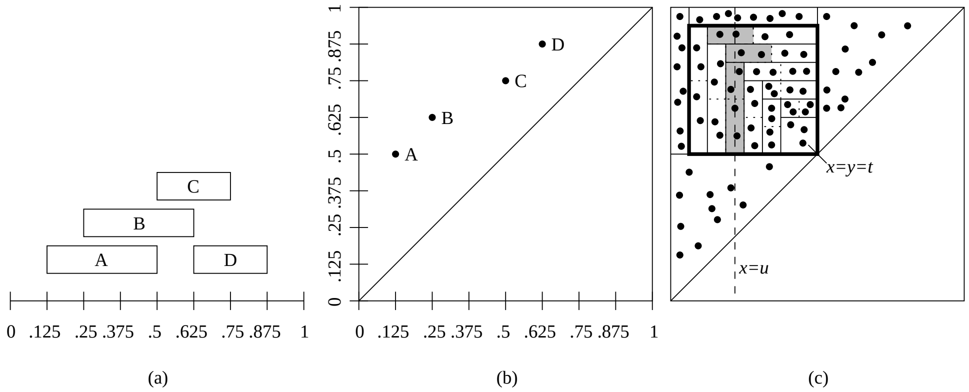

FIG. 2.1. *(a) Four jobs. (b) The jobs of (a) viewed as (x, y) points. Note that all jobs are above the $x = y$ diagonal. (c) Stripping a set of jobs with $H = 2$ and $\epsilon = .5$. The rectangle R_t defines the set Y of jobs. The first (leftmost) vertical and (topmost) horizontal strips each contain $H\lceil 1/\epsilon^2 \rceil = 8$ jobs. The remaining jobs, which comprise Y', are shown in the bold sub-rectangle. Y' is partitioned into strips containing $H\lceil 1/\epsilon \rceil = 4$ jobs each. Within each strip of 4 jobs in Y', the jobs are grouped (dotted lines) into groups of height $H = 2$. The groups that intersect the line $x = u$ are shaded.*

increasing starting x-coordinate in groups of size H (the last group of the last strip possibly smaller), and box them.

We analyze the construction using Figure 2.1(c). For all reals u, the jobs live at u correspond to the rectangle $R_u = \{(x, y) : x < u < y\}$. The fact that the jobs in Y are all live at t means that all the corresponding vertices are inside the rectangle R_t.

A box (other than a last box) corresponds to a group of exactly H jobs. If all are live at u, then these jobs contribute exactly H to $L_{Y'}(u)$, because each job has unit height, and the box contributes exactly H to $L_B(u)$. If none of them is live at u, then they contribute 0 to $L_{Y'}(u)$, and the box contributes 0 to $L_B(u)$. The troublesome case is the one in which some but not all of the jobs are live at u, since the jobs may contribute as little as 1 to $L_{Y'}(u)$, while the box still contributes H to $L_B(u)$.

Assume without loss of generality that $u < t$. Then the troublesome case corresponds to groups of jobs whose vertices are on both sides of the line $x = u$. By construction, this can happen to at most one group in each horizontal strip of Y'. (Recall that points in a horizontal strip are grouped in order of increasing x-coordinates.) Moreover, this can happen to groups inside at most one vertical strip of Y'.

Notice that if the line $x = u$ intersects, say, k horizontal strips of Y', then the rectangle R_u entirely contains at least $k - 1$ vertical strips of Y'. (In fact, in the current case, R_u entirely contains at least k vertical strips of Y'. In the symmetric case, when $t < u$, the number is $k - 1$. We argue without loss of generality.)

If the rectangle R_u does not contain any job of Y', then the lemma trivially holds. Otherwise, R_u must contain all the jobs in the (only) vertical strip of $Y - Y'$, which number $H\lceil 1/\epsilon^2 \rceil$.

Observe that

$$L_B(u) - L_{Y'}(u) \le kH + H\lceil 1/\epsilon \rceil + H, \tag{2.1}$$

where the first term accounts for the groups in the k horizontal strips that are intersected by $x = u$, the second term accounts for the groups in the single vertical strip that is intersected by $x = u$, and the third term accounts for (possibly) the last group of the last strip.

But

$$L_Y(u) \geq H\lceil 1/\epsilon^2\rceil + \max\{k-1,0\}\cdot H\lceil 1/\epsilon\rceil, \tag{2.2}$$

where the first term accounts for the vertical strip of $Y - Y'$, and the second accounts for the $\max\{k-1,0\}$ vertical strips of Y' that are contained in the rectangle R_u.

Now observe that

$$kH + H\lceil 1/\epsilon\rceil + H \leq (k-1)H + H/\epsilon + 3H; \tag{2.3}$$

$$H/\epsilon + 3H \leq H\epsilon(3/\epsilon + 1/\epsilon^2) \leq H\epsilon(4/\epsilon^2) \leq (4\epsilon)[H\lceil 1/\epsilon^2\rceil]; \tag{2.4}$$

$$(k-1)H \leq \epsilon\max\{k-1,0\}\cdot H\lceil 1/\epsilon\rceil \leq (4\epsilon)[\max\{k-1,0\}\cdot H\lceil 1/\epsilon\rceil]. \tag{2.5}$$

Combining equations (2.1)–(2.5) yields

$$L_B(u) - L_{Y'}(u) \leq 4\epsilon L_Y(u). \qquad \square$$

We call the jobs in $Y - Y'$ *unresolved jobs.*

2.2. Boxing Over All Times. We bootstrap Lemma 2.1 so that we can box all the jobs (i.e., not just jobs live at a particular, fixed x-coordinate) with just a small amount of wasted space. Our main technical result is particularly interesting when $L \gg (H\lg H)\lg(1/\epsilon)/\epsilon^2$. We use this theorem in the following sections to devise approximation schemes for Dynamic Storage Allocation.

Theorem 2.2. *Given a set Z of jobs, each of height 1, an integer* box-height parameter H, *and a sufficiently small positive ϵ, there exist a set B of boxes, each of height H, and a boxing of Z into B such that for all x-coordinates t,*

$$L_B(t) \leq (1+4\epsilon)L_Z(t) + O\left(\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right).$$

To prove Theorem 2.2, we are going to apply Lemma 2.1 many times, boxing the unresolved jobs into additional boxes as we go along. Our general goal is to keep the wasted load (free space) in those additional boxes small at any x-coordinate. We use the following recursive method. Given are

1. A set X of jobs and an open *bounding interval* I, such that $\forall j \in X, I_j \subseteq I$.
2. A nonempty finite set of *critical x-coordinates* $T = \{\inf I = t_0 < t_1 < \cdots < t_q < t_{q+1} = \sup I\} \subseteq I \cup \{\inf I, \sup I\}$.
3. A set F of *free spaces*. Each free space is an open sub-interval of I of height 1 *having endpoints in T*. Any free space $f \in F$ is called *spanning* if $f = I$ and *non-spanning* otherwise.

Initially, $X = Z$, $I = (0,1)$, $T = \{0,t,1\}$ for some arbitrary x-coordinate t at which some job from Z is live, and $F = \emptyset$. Recall that $I_j = (x_j, y_j)$ denotes the interval of job j. With the help of T, define partition

$$X = (R_1 \cup R_2 \cup \cdots \cup R_q) \cup (X_0 \cup X_1 \cup \cdots \cup X_q)$$

as follows. (See Figure 2.2.) First define $X_i = \{j \in X : I_j \subseteq (t_i, t_{i+1})\}$ for $0 \leq i \leq q$. The X_i's contain precisely the jobs that are not live at any critical time.

Then define the R_i's recursively like in an interval tree [2]. Define $X' = X \setminus (X_0 \cup X_1 \cup \cdots \cup X_q)$, the set of jobs that are live at some (not necessarily unique) critical time. Note that $q \geq 1$. Define $R_{\lceil q/2\rceil} = \{j \in X' : t_{\lceil q/2\rceil} \in I_j\}$. Define P to be the set

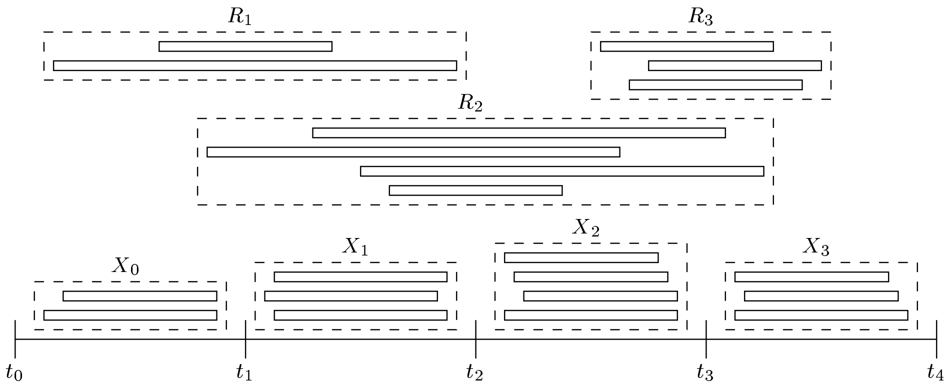

FIG. 2.2. *A set of jobs (solid rectangles) partitioned into subsets (dashed rectangles) X_i and R_j using critical x-coordinates $t_0, \ldots, t_4$.*

of remaining jobs j of X' with $y_j < t_{\lceil q/2 \rceil}$, and define Q to be the set of remaining jobs j of X' with $t_{\lceil q/2 \rceil} < x_j$. That is, $R_{\lceil q/2 \rceil}$ is the set of jobs that are live at $t_{\lceil q/2 \rceil}$; P is the set of jobs that are live at some critical time but end before $t_{\lceil q/2 \rceil}$; and Q is the set of jobs that are live at some critical time but start after $t_{\lceil q/2 \rceil}$. If $P \neq \emptyset$, recursively partition P using $\{t_1, t_2, \ldots, t_{\lceil q/2 \rceil - 1}\}$. Afterward, if $Q \neq \emptyset$, recursively partition Q using $\{t_{\lceil q/2 \rceil + 1}, t_{\lceil q/2 \rceil + 2}, \ldots, t_q\}$.

We will box the jobs in the R_i's, in the process establishing parallel, recursive instances of the decomposition to handle the X_i's. First, to each X_i associate a set F_i of intervals (free spaces), initially empty. As sections of free spaces in F are used to box jobs in the R_i's, the unused fragments will be deposited into the appropriate F_i's for use deeper in the recursion, to box jobs in the X_i's.

To box the jobs in the R_i's, first apply Lemma 2.1 to each R_i, $1 \leq i \leq q$, in any order; note that all jobs in R_i are live at t_i. For each i, this boxes all the jobs of R_i except for at most $2H\lceil 1/\epsilon^2 \rceil$ unresolved jobs. Now consider the set U of all the unresolved jobs from all the R_i's. Derive an optimal packing of U using interval graph coloring. (Recall that all jobs are of height one.) This packing has makespan L_U.

Let $s(F)$ denote the subset of spanning free spaces of F. If $|s(F)| < L_U$, create $\lceil (L_U - |s(F)|)/H \rceil$ boxes of height H and horizontal extent I. This yields $H\lceil (L_U - |s(F)|)/H \rceil$ new spanning free spaces; add them to F. Now there are at least as many spanning free spaces in F as rows of the packing of U.

For each $1 \leq j \leq L_U$, remove one spanning free space from F, and use it to place all the jobs in row j of the packing. This creates *gaps*, or unused portions, in the original free space, each of the form $[\alpha, \beta]$ where for some i, j: $t_i \leq \alpha < t_{i+1}$ and $t_j \leq \beta < t_{j+1}$; recall that $t_0 = \inf I$ and $t_{q+1} = \sup I$. For each such $[\alpha, \beta]$, if $i \neq j$ then split $[\alpha, \beta]$ into $(\alpha, t_{i+1}), (t_{i+1}, t_{i+2}), \ldots, (t_{j-1}, t_j), (t_j, \beta)$; and add (α, t_{i+1}) to F_i, (t_{i+1}, t_{i+2}) to F_{i+1}, …, (t_{j-1}, t_j) to F_{j-1}, and (t_j, β) to F_j. Otherwise ($i = j$), simply deposit (α, β) into F_i. This *fragments* the gaps.

Now all the jobs in all the R_i's are boxed. Consider the unused free spaces in F, if any. Each is of the form (t_i, t_j) for some $i \neq j$. Split each such (t_i, t_j) into $(t_i, t_{i+1}), (t_{i+1}, t_{i+2}), \ldots, (t_{j-1}, t_j)$. Add (t_i, t_{i+1}) to F_i, (t_{i+1}, t_{i+2}) to F_{i+1}, …, and (t_{j-1}, t_j) to F_{j-1}. This *passes down* the unused free spaces to the sub-problems.

In parallel for each $\ell = 0, 1, 2, \ldots, q$, if $X_\ell \neq \emptyset$, recursively apply the construction with new $X \leftarrow X_\ell$, new free space set $F \leftarrow F_\ell$, new bounding interval $I \leftarrow (t_\ell, t_{\ell+1})$, and new critical x-coordinate set $T \leftarrow \{\text{endpoints of elements of } F_\ell\} \cup \{t_\ell, t_{\ell+1}\}$. Note that this preserves the invariant that the endpoints of intervals in F are in T for each recursive subproblem. Also note that the unused free spaces passed down span their entire respective boundary intervals, whereas the gaps that were fragmented need not do so. This completes the construction.

2.3. Analysis. Fix a real u. (Recall that to prove Theorem 2.2 we want to bound the load of the boxes at any x-coordinate u in terms of the load of the original jobs at u.) For any m, define $I_m(u)$ to be the bounding interval of the depth-m recursive call whose bounding interval contains x-coordinate u. Similarly define $T_m(u)$ to be the set of critical x-coordinates, $p_m(u) = |T_m(u)|$, and $U_m(u)$ to be the set of unresolved jobs from all the R_i's in that same depth-m recursive call. Define $F_m(u)$ to be the set of free spaces during the same depth-m recursive call, and note that this set changes during that procedure. Figure 2.2, for example, depicts for some m a piece of the depth-m stage of the recursion; i.e., $I_m(u) = (t_0, t_4)$ for $u \in (t_0, t_4)$. The jobs in each X_i will form the set of jobs for each sub-problem at depth $m+1$; i.e., $I_{m+1}(u) = (t_0, t_1)$ for $u \in (t_0, t_1)$, $I_{m+1}(u) = (t_1, t_2)$ for $u \in (t_1, t_2)$, etc.

LEMMA 2.3. *For all m, $p_{m+1}(u) \le 2H\lceil 1/\epsilon^2\rceil \lceil \lg(p_m(u)+1)\rceil + 2$.*

Proof. Note that $p_0(u) = 3$. In the depth-m recursive call, the definition of the R_i's implies that the number of R_i's containing jobs with endpoints in $I_{m+1}(u)$ is at most $\lceil \lg(p_m(u)+1)\rceil$. (The process of repeatedly looking at the middle index resembles binary search, and the worst-case running time of binary search on a list of length p is $\lceil \lg(p+1)\rceil$.) For each such R_i, Lemma 2.1 is applied, and each application yields at most $2H\lceil 1/\epsilon^2\rceil$ unresolved jobs. Each unresolved job contributes at most one new endpoint and hence at most one critical x-coordinate to each of two different intervals $I_{m+1}(\cdot)$. The total number of critical x-coordinates contributed to $I_{m+1}(u)$ is thus at most $2H\lceil 1/\epsilon^2\rceil\lceil \lg(p_m(u)+1)\rceil + 2$, where the "+2" comes from the endpoints $\inf I_{m+1}(u)$ and $\sup I_{m+1}(u)$. ☐

COROLLARY 2.4. *For all m, $p_m(u) = O\left(\frac{H \lg H}{\epsilon^2} \lg \frac{1}{\epsilon}\right)$.*

Proof sketch. If $a_{m+1} \le k \lg a_m$ for all m, and $a_0 \le 3k \lg k$, then $a_m \le 3k\lg k$ for all m, a fact that is easily proven by induction on m. ☐

Let P denote the upper bound of the corollary. For all m, the number of critical x-coordinates in the subproblem defined by bounding interval $I_m(u)$ is at most P. Let $f_m(u)$ denote the number of free spaces in $F_m(u)$ *at the beginning* of that recursive call, and let $f'_m(u)$ denote the number of free spaces in $F_m(u)$ *at the end* of that recursive call. Abusing notation, let $L_m(u)$ denote the load of $U_m(u)$, the set of unresolved jobs from all the R_i's in the depth-m recursive call over bounding interval $I_m(u)$.

LEMMA 2.5. *For all m, $L_m(u) \le 2H\lceil 1/\epsilon^2\rceil\lceil \lg(P+1)\rceil$.*

Proof. As above, at most $\lceil \lg(P+1)\rceil$ R_i's include jobs that contain u' for any $u' \in I_m(u)$. By Lemma 2.1 each such R_i contributes at most $2H\lceil 1/\epsilon^2\rceil$ unresolved jobs to $U_m(u)$. ☐

COROLLARY 2.6. *For all m, $f'_m(u) \le f_m(u) + 2H\lceil 1/\epsilon^2\rceil\lceil \lg(P+1)\rceil$.*

Proof. Boxing the unresolved jobs in the depth-m recursive call requires creating at most $\lceil L_m(u)/H\rceil$ new boxes of height H, which adds at most $H\lceil L_m(u)/H\rceil$ new (spanning) free spaces to $F_m(u)$, after which free spaces are only removed from $F_m(u)$. Lemma 2.5 completes the proof. ☐

LEMMA 2.7. *For all m:*

1. $f_{m+1}(u) \le f_m(u) + 2H\lceil 1/\epsilon^2\rceil\lceil \lg(P+1)\rceil$.
2. *If* $f_m(u) \ge 4H\lceil 1/\epsilon^2\rceil\lceil \lg(P+1)\rceil$, *then* $f_{m+1}(u) \le f_m(u)$.

Proof. (1) follows from Corollary 2.6 and the fact that the fragmented gaps from any used spanning free space of $F_m(u)$ are distributed to distinct subproblems.

(2). The free spaces of $F_m(u)$ at the beginning of the depth-m recursive call come from two sources.

(i) Unused free spaces of $F_{m-1}(u)$ that were passed down; as noted above, they yield spanning free spaces of $F_m(u)$, i.e., free spaces that span the entire interval $I_m(u)$.

(ii) Fragmented free spaces obtained during the depth-$(m-1)$ recursive call by placing an unresolved job in a free space of $F_{m-1}(u)$. Lemma 2.5 implies that there are at most $2H\lceil 1/\epsilon^2\rceil\lceil\lg(P+1)\rceil$ such spaces.

Assume $f_m(u) \geq 4H\lceil 1/\epsilon^2\rceil\lceil\lg(P+1)\rceil$. The number of spanning free spaces is the total number of free spaces minus the number of non-spanning free spaces. Because there are at most $2H\lceil 1/\epsilon^2\rceil\lceil\lg(P+1)\rceil$ non-spanning free spaces, $F_m(u)$ starts with at least

$$4H\lceil 1/\epsilon^2\rceil\lceil\lg(P+1)\rceil - 2H\lceil 1/\epsilon^2\rceil\lceil\lg(P+1)\rceil = 2H\lceil 1/\epsilon^2\rceil\lceil\lg(P+1)\rceil$$

spanning free spaces. When the unresolved jobs are boxed during the depth-m recursive call, there are enough spanning free spaces to fit *all* the unresolved jobs (again by Lemma 2.5), so no new free spaces are created. Thus, $f_{m+1}(u) \leq f_m(u)$. □

COROLLARY 2.8. *For all m, $f_m(u) = O\left(\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right)$.*

Proof. Follows from Corollary 2.4 and Lemma 2.7. □

COROLLARY 2.9. *For all m, $f'_m(u) = O\left(\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right)$.*

Proof. Follows from Corollaries 2.4, 2.6, and 2.8. □

We can now finish the proof of Theorem 2.2. Consider any x-coordinate u, and let m' denote the greatest depth of the recursion for which there existed an $I_{m'}(u)$. Of the jobs in Z that are live at u, let Z_R denote those that are resolved during applications of Lemma 2.1, and let Z_U denote the rest. The load at u of the boxes used to resolve jobs in Z_R is at most $L_{Z_R}(u) + 4\epsilon L_Z(u)$, by Lemma 2.1. The load at u of the boxes used to place jobs in Z_U is $L_{Z_U}(u) + f'_{m'}(u)$, because jobs in Z_U are only placed into free spaces in the various $F_m(u)$'s, and unused free spaces containing u in $F_m(u)$ for any $0 \leq m < m'$ are inherited by $F_{m+1}(u)$. Therefore, using Corollary 2.9 and the fact that $L_Z(u) = L_{Z_R}(u) + L_{Z_U}(u)$, the total storage allocated at x-coordinate u does not exceed

$$\left[L_{Z_R}(u) + 4\epsilon L_Z(u)\right] + \left[L_{Z_U}(u) + f'_{m'}(u)\right] = (1+4\epsilon)L_Z(u) + O\left(\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right). \quad \square$$

3. Dynamic Programming Solutions. In addition to boxing, our later results use the following results based on dynamic programming to solve simple cases.

THEOREM 3.1. *The optimal makespan can be determined in $O(\mathrm{poly}(n)(3L)^{2L+1})$ time.*

Proof. First, using the fact that $OPT \leq 3L$ [5], guess the optimal makespan M^*. Build an array T with, for each x, an entry for each feasible placement C_x of jobs that cross the vertical line at x. Now define $T[C_x]$ to be 0-1, with $T[C_x] = 1$ if and only if the set of jobs that intersect $[0, x)$ can be placed using height at most M^* with a placement that respects C_x. We have $T[C_x] = 1$ if and only if there is a $C_{x-1} = 1$ such that C_{x-1} is compatible with C_x and $T[C_{x-1}] = 1$.

There are $3L$ possibilities for M^*. There are $2n$ possibilities for x. For each x, there are $(M^*)^L \leq (3L)^L$ configurations C_x. Computing $T[C_x]$ takes time at most

$(3L)^L$. The overall time bound is thus

$$O((3L)(3L)^L(3L)^L\mathrm{poly}(n)),$$

which is $O(\mathrm{poly}(n)(3L)^{2L+1})$. ∎

COROLLARY 3.2. DYNAMIC STORAGE ALLOCATION *can be solved optimally in polynomial time when* $L = O(\lg n/\lg\lg n)$.

Proof. Follows immediately from Theorem 3.1. ∎

THEOREM 3.3. *Let C be a positive real and let $\alpha = \alpha(n)$ be a positive function of n such that for all sufficiently large n, $\alpha(n) \geq \lg\lg n/(C\lg n)$. There is a PTAS for the special case of* DYNAMIC STORAGE ALLOCATION *in which* $h_{\min}/L \geq \alpha(n)$.

Proof. Given C, α, and $\epsilon > 0$, apply the dynamic program in the proof of Theorem 3.1 to the same jobs, except with height h replaced by $\lceil h/(\epsilon\alpha(n)L)\rceil$. Let L' be the load in the new problem. If there were no ceiling in the definition of the new heights, the load in the new problem would be $L/(\epsilon\alpha(n)L) = 1/(\epsilon\alpha(n))$. Since each original height h satisfies $h/(\epsilon\alpha(n)L) \geq 1/\epsilon$, the ceiling introduces an additional factor of at most $1+\epsilon$. It follows that $L' \leq (1+\epsilon)/(\epsilon\alpha(n)) \leq (2C/\epsilon)\lg n/\lg\lg n$ if n is large enough and $\epsilon \leq 1$. The new problem can be solved exactly in polynomial time by Corollary 3.2. ∎

4. Algorithm for Bounded-Height Jobs. Let X be the set of all jobs and ϵ be a sufficiently small positive real. Recall that L is a trivial lower bound to OPT. If all the jobs have the same height, then the problem reduces to interval graph coloring and can be solved optimally in a greedy fashion; furthermore, $OPT = L$. Our algorithm will be a reduction to this simple case. Let $H = h_{\max}\lceil 1/\epsilon\rceil$, and assume that the maximum height of a job, $h_{\max}$, is a constant.

ALGORITHM:

1. If $L \leq C\frac{1}{\epsilon^4}\lg^2\frac{1}{\epsilon}$, for some C to be determined later, apply Corollary 3.2 and halt.

2. For each $h = 1, 2, 3, \ldots, h_{\max}$:

(i) Let X_h denote the set of jobs of height h. Let $H_h = \lfloor H/h\rfloor h$.

(ii) Scale each job in X_h down by a factor of h, apply Theorem 2.2 with box-height parameter $\lfloor H/h\rfloor$ and the given ϵ to this new set of jobs, and then scale the jobs back up by the same factor h.

3. Enlarge the boxes so that they all have height exactly H. Call the new set of boxes B'.

4. Apply the greedy algorithm for interval graph coloring to B'.

THEOREM 4.1. *The makespan of the packing produced by the algorithm is at most $(1+15\epsilon)OPT$. (Recall that $h_{\max}$ is a constant and ϵ is sufficiently small.)*

Proof. Assume for now that the algorithm does not stop in step (1). We argue for all t. By Theorem 2.2, step (2) produces a boxing B_h of the jobs of X_h into boxes of height H_h such that

$$L_{B_h}(t) \leq (1+4\epsilon)L_{X_h}(t) + O\left(\frac{1}{\epsilon^2}\cdot\frac{H}{h}\left(\lg\frac{H}{h}\right)\lg\frac{1}{\epsilon}\right)h. \tag{4.1}$$

(The final h comes from the scaling back up at the end of step (2); we have applied the theorem to jobs of height 1 with box-height parameter there equal to $\lfloor H/h\rfloor = H_h/h$.) Let $B = B_1 \cup B_2 \cup \cdots \cup B_{h_{\max}}$. Adding equation (4.1) yields

$$L_B(t) \leq (1+4\epsilon)L_X(t) + O\left(h_{\max}\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right).$$

Enlarging the boxes increases them by at most a factor of $1/(1-\epsilon)$. Altogether,

$$\begin{aligned}\text{makespan} &= \max_t L_{B'}(t)\\ &\le \max_t \left[\frac{1}{1-\epsilon}\left((1+4\epsilon)L_X(t) + O\left(h_{\max}\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right)\right)\right]\\ &\le \frac{1+4\epsilon}{1-\epsilon}\max_t L_X(t) + \frac{1}{1-\epsilon}O\left(h_{\max}\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right).\end{aligned}$$

$$\begin{aligned}\text{Therefore, makespan} &\le \frac{1+4\epsilon}{1-\epsilon}L + O\left(h_{\max}\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right) \qquad (4.2)\\ &\le \frac{1+4\epsilon}{1-\epsilon}OPT + O\left(h_{\max}\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right).\end{aligned}$$

Choose C so that the error term $O\left(h_{\max}\frac{H\lg H}{\epsilon^2}\lg\frac{1}{\epsilon}\right) \le \epsilon L$, which is possible under the assumptions that $L > C\frac{1}{\epsilon^4}\lg^2\frac{1}{\epsilon}$, $h_{\max}$ is a constant, and $H = h_{\max}\lceil 1/\epsilon\rceil$. If $L \le C\frac{1}{\epsilon^4}\lg^2\frac{1}{\epsilon}$, then the algorithm stops in step (1), and Theorem 3.1 applies. Otherwise, it follows that makespan $\le (1+15\epsilon)OPT$, because $L \le OPT$. □

THEOREM 4.2. *The makespan produced by the packing is* $(1+O(((\lg^2 L)/L)^{1/4}))L$. *(Recall that $h_{\max}$ is a constant.)*

Proof. Set $\epsilon = \sqrt{\lg L}/L^{1/4}$, and run the algorithm starting in step (2). The claim follows from inequality (4.2). □

5. Algorithms for Larger Jobs. From Theorem 2.2, we can prove the following corollary.

COROLLARY 5.1. *Let H be a positive integer* box-height parameter*, $h_{\min}$ be a positive real, and $\epsilon > 0$ be a sufficiently small* error parameter. *Given a set Z of jobs, each of height between $h_{\min}$ and ϵH, there exist a set B of boxes, each of height H, and a boxing of Z into B such that for all x-coordinates t,*

$$L_B(t) \le (1+9\epsilon)L_Z(t) + O\left(\frac{H\lg^2(H/h_{\min})}{\epsilon^4}\right).$$

Proof. We construct an appropriate boxing. First, round the job heights: each height h is rounded up to $\lfloor(1+\epsilon)^i\rfloor$, where i is defined by $(1+\epsilon)^{i-1} < h \le (1+\epsilon)^i$. Let Y denote the resulting set of rounded jobs.

Now partition the jobs according to their heights. For each rounded height h, let Y_h denote the set of jobs of height h. Divide the heights of all jobs in Y_h by h; apply Theorem 2.2 with box-height parameter $\lfloor H/h\rfloor$; and then multiply all box heights by h to get a set B_h of boxes of height at most H. The output is a set $B = \bigcup_h B_h$ of boxes, which we can assume are all of height H.

Let us analyze this construction. There are approximately $\log_{(1+\epsilon)}(\epsilon H/h_{\min}) = O(\lg(H/h_{\min})/\epsilon)$ parts in the partition. Applying Theorem 2.2 to Y_h yields

$$\begin{aligned}\forall t,\quad L_{B_h}(t) &\le (1+4\epsilon)L_{Y_h}(t) + O\left(h\frac{\lfloor H/h\rfloor\lg\lfloor H/h\rfloor}{\epsilon^2}\lg\frac{1}{\epsilon}\right)\\ &\le (1+4\epsilon)L_{Y_h}(t) + O\left(\frac{H\lg(H/h_{\min})}{\epsilon^3}\right),\end{aligned}$$

because $h_{\min} \le h$ and $\lg(1/\epsilon) \le 1/\epsilon$. Summing over h, we get

$$\forall t,\quad L_B(t) \le (1+4\epsilon)L_Y(t) + O\left(\frac{H\lg^2(H/h_{\min})}{\epsilon^4}\right).$$

Since rounding increased heights by a factor of $1+\epsilon$ at most, we have $L_Y(t) \le (1+\epsilon)L_Z(t)$. Since $(1+\epsilon)(1+4\epsilon) \le 1+9\epsilon$ for $\epsilon \le 1$, the corollary follows. ∎

We are now ready to prove our main theorem.

THEOREM 5.2. *For any* fixed $\epsilon \in (0,1]$ *there exists a polynomial-time algorithm, which depends on ϵ, that takes an arbitrary set X of jobs as input and produces a feasible solution to* DYNAMIC STORAGE ALLOCATION *on X with makespan at most $(1+c\epsilon)L + O(h_{\max}/\epsilon^6)$, where c is a universal constant.*

Proof. It suffices to prove the theorem for $\epsilon \le \epsilon_0$ for some small, positive ϵ_0, for the following reason. Suppose for all $\epsilon \le \epsilon_0$ we get a makespan bound of $(1+c_0\epsilon)L + c_1(h_{\max}/\epsilon^6)$ for some constants c_0 and c_1. Then for $\epsilon \in (\epsilon_0, 1]$ we simply use $(1+c_0\epsilon_0)L + c_1 h_{\max}/\epsilon_0^6 \le (1+c_0\epsilon)L + (c_1/\epsilon_0^6)h_{\max}/\epsilon^6$. Similarly, by reducing ϵ by a factor of at most two, we may assume that $1/\epsilon \in \mathbb{Z}$.

We are going to apply Corollary 5.1 repeatedly, boxing the smallest jobs so as to increase the minimum job height $h_{\min}$ until it gets close enough to the maximum job height $h_{\max}$ that we can finish with a last application of Corollary 5.1.

We argue for all t. Let r denote the ratio $h_{\max}/h_{\min}$, and recall that $L = L_X$. Assume first that $\lg^2 r \ge 1/\epsilon$, and set $\mu = \epsilon/\lg^2 r$ and $H = \lceil \mu^5 h_{\max}/\lg^2 r\rceil$. Consider the partition $X = X_s \cup X_\ell$, where X_s denotes the jobs of height at most μH and $X_\ell = X \setminus X_s$. Now apply Corollary 5.1 to X_s with box-height parameter H and error parameter μ. This yields a set B_s of boxes of height H into which the jobs of X_s fit such that for some constants c_2 and c_3,

$$\begin{aligned}
L_{B_s}(t) &\le (1+9\mu)L_{X_s}(t) + O\left(\frac{H\lg^2(H/h_{\min})}{\mu^4}\right)\\
&\le (1+9\mu)L_{X_s}(t) + c_2\mu h_{\max}\\
&\le L_{X_s}(t) + c_3\mu L(t)\\
&= L_{X_s}(t) + (c_3\epsilon/\lg^2 r)L(t).
\end{aligned}$$

Now consider B_s as a set of jobs and the revised problem on $X' = B_s \cup X_\ell$. For this new problem, the load $L'(t)$ is at most $(1+c_3\epsilon/\lg^2 r)L(t)$. Moreover, the new minimum height $h'_{\min}$ is at least μH, and the maximum height remains at most $h_{\max}$, so we get the new ratio

$$r' \le \frac{h_{\max}}{h'_{min}} \le \frac{h_{\max}}{\mu\lceil \mu^5 h_{\max}/\lg^2 r\rceil} \le \frac{h_{\max}}{\mu^6 h_{\max}/\lg^2 r} = \frac{\lg^2 r}{\mu^6} = \frac{\lg^{14} r}{\epsilon^6} \le \lg^{26} r.$$

Recall that the above construction was conditioned on $\lg^2 r \ge 1/\epsilon$. For ϵ sufficiently small, this implies $\lg^{26} r \le \sqrt{r}$ and hence that $r' \le \sqrt{r}$ and also $\lg^2 r' \le \lg^2 \sqrt{r} \le (1/4)\lg^2 r$.

Iterate the above boxing of small jobs, each time using new error parameter $\mu' = \epsilon/\lg^2 r'$, until it yields a problem X^* with minimum job height $h^*_{\min}$ for which the ratio $r^* = h_{\max}/h^*_{\min}$ is such that $\lg^2 r^* < 1/\epsilon$. Since the ratio decreases by at least a square root each time, this process terminates in polynomial time.

Let $(r_0, ..., r_p = r^*)$ be the sequence of ratios, and let $(L_0(t), ..., L_p(t) = L^*(t))$ be the sequence of loads. For $0 \le i < p$ we know that

$$L_{i+1}(t) \le (1 + c_3\epsilon/\lg^2 r_i)L_i(t); \tag{5.1}$$

$$\lg^2 r_{i+1} \le (1/4)\lg^2 r_i. \tag{5.2}$$

For some constant c_4, we therefore get

$$\begin{aligned} L^*(t) &\le L_0(t) \prod_{i=0}^{p-1} (1 + c_3\epsilon/\lg^2 r_i), \quad \text{by (5.1)} \\ &\le L_0(t) \exp\left(\sum_{i=0}^{p-1} (c_3\epsilon/\lg^2 r_i) \right) \\ &\le L_0(t) \exp(c_4\epsilon/\lg^2 r_{p-1}), \quad \text{by (5.2)} \\ &\le L_0(t) \exp(c_4\epsilon^2), \quad \text{because } \lg^2 r_{p-1} \ge 1/\epsilon \\ &\le (1 + 2c_4\epsilon^2) L_0(t), \quad \text{for } \epsilon \text{ sufficiently small.} \end{aligned}$$

Now apply Corollary 5.1 to all of X^* with box-height parameter $H = h_{\max}/\epsilon$ (recall that $1/\epsilon \in \mathbb{Z}$) and error parameter ϵ; this is the "last application" of Corollary 5.1 to which we alluded earlier. For some constant c, this yields a final set X^f of jobs of identical height H and with load

$$\begin{aligned} L^f(t) &= (1 + 9\epsilon)L^*(t) + O\left(\frac{H \lg^2(H/h^*_{\min})}{\epsilon^4} \right) \\ &\le (1 + 9\epsilon)(1 + 2c_4\epsilon^2)L_0(t) + O\left(\frac{H \lg^2(H/h^*_{\min})}{\epsilon^4} \right) \\ &\le (1 + c\epsilon)L_0(t) + O\left(\frac{h_{\max} \lg^2(r^*/\epsilon)}{\epsilon^5} \right). \end{aligned}$$

Noting that $\lg^2 r^* < 1/\epsilon$ and $\lg^2(1/\epsilon) < 1/\epsilon$ for ϵ sufficiently small, we derive $\lg^2(r^*/\epsilon) \le 4/\epsilon$. Therefore

$$L^f(t) \le (1 + c\epsilon)L_0(t) + O(h_{\max}/\epsilon^6). \qquad \square$$

COROLLARY 5.3. *There exists a polynomial-time algorithm that takes an arbitrary set X of jobs as input and produces a feasible solution to* DYNAMIC STORAGE ALLOCATION *on X with makespan at most* $(1 + O((h_{\max}/L)^{1/7}))L$.

Proof. Apply Theorem 5.2 to X with $\epsilon = (h_{\max}/L)^{1/7}$. $\square$

COROLLARY 5.4. *Fix some function $f(m) = o(m)$. Then there is a PTAS for* DYNAMIC STORAGE ALLOCATION *when $h_{\max} \le f(L)$.*

Proof. Given $\epsilon > 0$, there is an L_0 such that $f(L) \le \epsilon^7 L$ for all $L \ge L_0$. If $L < L_0$, use the dynamic program of Theorem 3.1. If $L \ge L_0$, then

$$h_{\max} \le f(L) \le \epsilon^7 L.$$

Apply the algorithm of Theorem 5.2. The error term of Theorem 5.2 is $O(h_{\max}/\epsilon^6) \le c'\epsilon L$ for some constant c'. Hence the makespan of the schedule is at most $(1 + (c + c')\epsilon)L$. $\square$

THEOREM 5.5. *For all $\epsilon > 0$, there exists a polynomial-time $(2+\epsilon)$-approximation algorithm for* DYNAMIC STORAGE ALLOCATION.

Proof. Consider some small positive δ to be determined later. Let $X = X_s \cup X_\ell$, where X_s is the set of jobs of height less than $\delta^7 L$ and $X_\ell = X \setminus X_s$. Use Theorem 5.2 with error parameter δ to pack the jobs in X_s, yielding a $(1 + c'\delta)$-approximation for some constant c'. Apply the $(1+\delta)$-approximation algorithm implied by Theorem 3.3 with the same δ to pack the jobs in X_ℓ, which is possible because the load divided by the minimum height is at most $1/\delta^7$, which is certainly at most $C \lg n/\lg\lg n$ for $C = 1/\delta^7$; this yields a $(1 + \delta)$-approximation. Choose δ so that $\delta(c' + 1) = \epsilon$. $\square$

6. Lower Bounds For $OPT-LOAD$**.** Fix a positive integer d. Given a bipartite multigraph $G = (X, Y, E)$, $X = Y = \{1, 2, \ldots, n\}$, build an instance of Dynamic Storage Allocation as follows. Start with n rectangles of height d, with $x_i = 0$, $y_i = i$, for $i = 1, 2, \ldots, n$, and then add n more rectangles of height d, with $x_i = 3n - i$, $y_i = 3n$, for $i = 1, 2, \ldots, n$. Call these $2n$ items *d-items*. For each $e = (j, k) \in E$ (with multiplicity, as E is a multiset), $1 \le j, k \le n$, add one item of height 1 having $x_i = j$, $y_i = 3n - k$. Call these jobs *1-items*. These are all the jobs. Call the instance I.

Lemma 6.1. *If G is d-regular, then $L_I(t) \le dn$ for all $t \in (0, 3n)$.*

Proof. For any non-integer $t \in [n, 2n]$, the statement of the lemma is obvious, since the set of jobs that intersect $(n, 2n)$ is the set of all 1-items, that set has size dn, and each such job has height 1.

For any non-integer $t \in [0, n]$, let $j = \lfloor t \rfloor$. Crossing t are $n - j$ d-items (these jobs have $x_i = 0$) and dj 1-items (these have $x_i \in \{1, 2, \ldots, j\}$), because G is d-regular, and no others. Hence the load at t is $(n - j) \cdot d + (dj) \cdot 1 = dn$.

For a non-integer $t \in [2n, 3n]$, the argument is symmetric to the case of $t \in [0, n]$.

For any integer $t \in (0, 3n)$, $L_I(\cdot)$ achieves a local minimum at t. ∎

Lemma 6.2. *For any positive integers n, d, f, and $s < \frac{n^{1/2-1/f}}{\sqrt{d}}$, there exists a d-regular bipartite multigraph (X, Y, E) with $X = Y = \{1, \ldots, n\}$ such that for any subsets X' of s consecutive vertices from X and Y' of s consecutive vertices from Y, the number of edges induced by $X' \cup Y'$ is smaller than f.*

Proof. Consider the following probability space of n-vertex-by-n-vertex d-regular bipartite multigraphs. Choose d independent permutations π of $\{1, \ldots, n\}$; for each, add edges between $i \in X$ and $\pi(i) \in Y$ for $1 \le i \le n$. We will show that the probability that a graph not of the claimed structure exists is less than 1; the claim then follows. Consider such a graph $G = (X, Y, E)$. There are some j and k such that $X' = \{j, \ldots, j+s-1\} \subseteq X$, $Y' = \{k, \ldots, k+s-1\} \subseteq Y$, and the subgraph induced by $X' \cup Y'$ has at least f edges. Consider any f such edges and fix an arbitrary order of them; this forms a sequence of edges $F = ((u_1, v_1), \ldots, (u_f, v_f))$ where each $u_i \in X'$ and each $v_i \in Y'$. There are no more than n^2 choices for the pair (j, k) and no more than s^{2f} possible sequences F. The probability that any particular edge exists is at most d/n. By the union bound, the probability that G exists is therefore no more than $P = n^2 s^{2f} (d/n)^f$. Simple algebra yields $P < 1$ when $s < \frac{n^{1/2-1/f}}{\sqrt{d}}$. ∎

Theorem 6.3. *For all positive integers n and d, there is an instance of* Dynamic Storage Allocation *with maximum job height d, $L(t) \le dn$ for all t, and $OPT - LOAD \ge \frac{n^{1/2-1/\lceil d/2 \rceil}}{12\sqrt{d}}$.*

Proof. Let $G = (X, Y, E)$ be a d-regular bipartite multigraph such that $|X| = |Y| = n$. Build the instance I associated with G, as above, with $2n$ d-items and dn 1-items. Lemma 6.1 shows that the load is at most dn everywhere.

Assume an optimal solution, and denote by Z the region defined by the isothetic bounding box of the d-items. Z is the rectangle $[0, 3n] \times [\min, \max]$, where min is the minimum y-coordinate of the bottoms of the $2n$ d-items and max is the maximum y-coordinate of the tops of the $2n$ d-items. Define Z' to be the set of 1-items placed outside Z. Because $\max - \min \ge dn$ and the load is at most dn everywhere, it follows that the load of Z' lower bounds $OPT - LOAD$ and that the area of the jobs in Z' lower bounds the empty space inside Z.

Now, for any positive real $s < n$, assume that fewer than $s/12$ 1-items are placed outside Z and also that $\max - \min < dn + s/12$. The area of Z is thus $(\max - \min)(3n) < (dn + s/12)(3n)$. The total area of all the jobs is $(dn)(3n)$. Therefore, if all the jobs were placed in Z, the empty space in Z would be at most $(s/12)(3n)$. For each job placed outside Z, the empty space inside Z grows by at most $3n$. The total empty space inside Z is thus less than $2(s/12)(3n) = sn/2$.

Consider a d-item to be comprised of d unit-height *rows*, which are placed contiguously in the plane. Define the *left* (resp., *right*) *d-items* to be those whose left (resp., right) endpoints are 0 (resp., $3n$). Call region $(3n-1, 3n) \times (k, k+1)$ for any integer $k \in [\min, \max - 1]$ a *right gap* if it is unoccupied by every right d-item. Clearly $OPT - LOAD$ is at least the number of right gaps, so we may assume this number is less than $n/2$. It follows that at least $n/2$ left d-items are adjacent to no right gaps; i.e., for each such left d-item x, there is a row of some right d-item to the right of each one of x's d rows. Call these the *good left d-items.*

For any left d-item J, define its *neighbor right d-items* to be the (at most two) right d-items whose y-coordinates overlap with J, and call the actual rows of those neighbors that occupy the same y-coordinates as J its *neighbor right rows.* Because there are at least $n/2$ good left d-items and the total empty space inside Z is less than $sn/2$, there must be at least one good left d-item J with less than s empty space between it and its neighbor right rows. Because $s < n$, there is some 1-item x placed between each row of J and the corresponding neighbor right row; furthermore, the left endpoint of x is within $s-1$ of the right endpoint of J, and the right endpoint of x is within $s-1$ of the left endpoint of the corresponding neighbor right row. For one of the at most two neighbor right d-items K of J, there are at least $\lceil d/2 \rceil$ neighbor right rows of J belonging to K. In G, therefore, there are subsets of $\lfloor s \rfloor$ consecutive vertices of X (starting at vertex $j \in X$, where j is the right endpoint of J) and $\lfloor s \rfloor$ consecutive vertices of Y (starting at vertex $k - \lfloor s \rfloor + 1 \in Y$, where k is the left endpoint of K) with at least $\lceil d/2 \rceil$ edges between them, corresponding to these 1-items.

If $s < \frac{n^{1/2 - 1/\lceil d/2 \rceil}}{\sqrt{d}}$, Lemma 6.2 (with $f = \lceil d/2 \rceil$) shows that there is some G' without such a window of edges. By contradiction, therefore, for any such G' there must be at least $s/12$ 1-items placed outside Z, or else $\max - \min \geq s/12$; in either case, $OPT - LOAD \geq s/12$. The theorem follows by choosing $s = \frac{n^{1/2 - 1/\lceil d/2 \rceil}}{\sqrt{d}}$ and noting that in fact $OPT - LOAD \geq \lceil s/12 \rceil$. ∎

Corollary 6.4. *For any $\epsilon > 0$, there exist $c, d > 0$ such that for all sufficiently large integers L, there is an instance of* Dynamic Storage Allocation *with maximum job height d, $L(t) \leq L$ for all t, and $OPT - LOAD \geq cL^{1/2-\epsilon}$.*

Proof. Fix $d = \lceil 2/\epsilon \rceil$, and choose a large $L \in \mathbb{Z}$. Define $n, r \in \mathbb{Z}$ such that $n > 0$, $0 \leq r < d$, and $L = nd + r$. Define $L' = nd$. By Theorem 6.3, there is a Dynamic Storage Allocation instance I with maximum job height d, $L_I(t) \leq L' = dn$ for all t, and $\Delta = OPT - LOAD \geq \frac{n^{1/2-1/\lceil d/2 \rceil}}{12\sqrt{d}}$. Because $n = \frac{L-r}{d} \leq \frac{L}{d}$, it follows that $L_I(t) \leq L$ for all t; and because $n \geq \frac{L}{2d}$, it follows that

$$\Delta \geq \frac{n^{1/2-1/\lceil d/2 \rceil}}{12\sqrt{d}} \geq \frac{\left(\frac{L}{2d}\right)^{1/2-1/\lceil d/2 \rceil}}{12\sqrt{d}} = \frac{1}{12\sqrt{d}} \left(\frac{1}{2d}\right)^{1/2-1/\lceil d/2 \rceil} L^{1/2-1/\lceil d/2 \rceil}.$$

The claim follows by setting

$$c = \frac{1}{12\sqrt{d}} \left(\frac{1}{2d}\right)^{1/2-1/\lceil d/2 \rceil}$$

and noting that $1/\lceil d/2 \rceil \leq \epsilon$. ∎

Corollary 6.5. *For any $\epsilon > 0$, there exists a $c' > 0$ such that for all sufficiently large integers $h_{\max}$, there is a* Dynamic Storage Allocation *instance* *with maximum job height $h_{\max}$, maximum load at most L, and $OPT - LOAD \geq c'(h_{\max}/L)^{1/2+\epsilon}L$, for infinitely many integers L.*

Proof. Given $\epsilon > 0$, define $h = h(\epsilon)$ and $c > 0$ so that for all sufficiently large L_0, Corollary 6.4 yields a Dynamic Storage Allocation instance with maximum job height h, maximum load at most L_0, and optimum makespan $OPT_0 \geq L_0 + cL_0^{1/2-\epsilon}$. Set $c' = c/h^{1/2+\epsilon}$. For all integers L and $h_{\max}$ such that L and $L/h_{\max}$ are sufficiently large and $h_{\max}/h$ and $L/(h_{\max}/h)$ are integral, set $L_0 = L/(h_{\max}/h)$. Now scale up the instance by $h_{\max}/h$. The new optimum makespan is $OPT = (h_{\max}/h)OPT_0$. The new load is at most $L_0 h_{\max}/h = L$, and

$$\begin{aligned}
OPT &= (h_{\max}/h)OPT_0 \\
&\geq (h_{\max}/h)(L_0 + cL_0^{1/2-\epsilon}) \\
&= L + (h_{\max}/h)cL_0^{1/2-\epsilon} \\
&= L(1 + (h_{\max}/(hL))cL_0^{1/2-\epsilon}) \\
&= L(1 + (h_{\max}/(hL))c(hL/h_{\max})^{1/2-\epsilon}) \\
&= L(1 + (c/h^{1/2+\epsilon})(h_{\max}/L)^{1/2+\epsilon}) \\
&= L(1 + c'(h_{\max}/L)^{1/2+\epsilon}). \qquad \square
\end{aligned}$$

Acknowledgments. We thank the anonymous referees for several helpful comments.

REFERENCES

[1] M. Chrobak and M. Slusarek, *On some packing problem related to dynamic storage allocation*, RAIRO Informatique Theorique et Applications, 22 (1988), pp. 487–499.

[2] M. de Berg, M. van Krevald, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, Berlin, second ed., 2000.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[4] J. Gergov, *Approximation algorithms for dynamic storage allocation*, in Proc. 4th European Symposium on Algorithms, vol. 1136 of Lecture Notes in Computer Science, Barcelona, Spain, 1996, Springer-Verlag, pp. 52–61.

[5] ———, *Algorithms for compile-time memory optimization*, in Proc. 10th ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 1999, SIAM, pp. S907–908.

[6] H. A. Kierstead, *The linearity of first-fit coloring of interval graphs*, SIAM J. Discrete Mathematics, 1 (1988), pp. 526–530.

[7] ———, *A polynomial time approximation algorithm for dynamic storage allocation*, Discrete Mathematics, 88 (1991), pp. 231–237.

[8] D. R. Woodall, *Problem no. 4*, in Proc. British Combinatorial Conference (1973), vol. 13 of London Mathematical Society Lecture Notes Series, Cambridge University Press, 1974, p. 202.