TIGHT BOUNDS FOR DYNAMIC STORAGE ALLOCATION *

MICHAEL G. LUBY †, JOSEPH (SEFFI) NAOR ‡, AND ARIEL ORDA §

Abstract. This paper is concerned with on-line storage allocation to processes in a dynamic environment. This problem has been extensively studied in the past. We provide a new, tighter bound, for the competitive ratio of the well known First Fit algorithm. This bound is obtained by considering a new parameter, namely the maximum number of concurrent active processes. We observe that this bound is also a lower bound on the competitive ratio of any deterministic on-line algorithm. Our second contribution is an on-line allocation algorithm which uses coloring techniques. We show that the competitive ratio of this algorithm is the same as that of First Fit. Furthermore, we indicate that this algorithm may be advantageous in certain applications. Our third contribution is to analyze the performance of randomized algorithms for this problem. We obtain lower bounds on the competitive ratio which are close to the best deterministic upper bounds.

Key words. on-line algorithms, memory management, dynamic storage allocation, bandwidth allocation, first-fit, interval graph.

AMS subject classifications. 60C05, 60E15, 68Q25, 68R10.

1. Introduction.

This paper is concerned with a classic problem in computer science: the allocation of area in a one dimensional storage device to processes in a dynamic environment. In a typical setting, at the time of arrival of a process, it is allocated storage area. This area is required to form a contiguous location in the storage device. Once allocated, a process cannot be moved to a different location. At some later point in time (*unknown* at the time of allocation), the process leaves, thereby liberating the storage area it occupied and making it available to other processes. As a result, wasted space or "holes" are generated over time in the storage device. The objective is to find an allocation algorithm that minimizes the wasted space. This is a typical on-line setting in which decisions must be based upon the current state without knowledge of future events.

The processes form an interval graph denoted by $G = (V, E)$; a process corresponds to the interval defined (on the time axis) between its arrival and departure time. Each interval is associated with a weight which is equal to the storage area required by the corresponding process. The allocation of storage area can be thought of as a *weighted coloring* of G: the color of interval i is a range $[a_i, b_i]$, (where a_i and b_i are integers), such that $b_i - a_i + 1$ is equal to the weight of interval i, and there is no intersection between the ranges of two intervals that overlap. The objective is to minimize b_i, where b_i is taken over all intervals.

We denote the maximum weight, taken over all intervals, by $W_{\max}$. Let $\omega^*(G)$ denote the weight of the heaviest clique in G, where the weight of a clique is the sum

* A preliminary version of this paper appeared in the proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, Virginia, (1994), pp. 724-732.

† International Computer Science Institute, University of California, Berkeley, CA 94704 (luby@icsi.berkeley.edu). Research supported in part by NSF Grant CCR-9016468 and Grant No. 89-00312 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

‡ Department of Computer Science, Technion, Haifa 32000, Israel (naor@cs.technion.ac.il). Research supported in part by Grant No. 92-00225 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and by the Technion Argentinian Research Fund. Part of this research was done while the author was visiting DIMACS, Rutgers University, NJ.

§ Department of Electrical Engineering, Technion, Haifa 32000, Israel (ariel@ee.technion.ac.il).

of the weights of the intervals in the clique. Clearly, $\omega^*(G)$ is a lower bound on the area used by *any* allocation algorithm.

We note that the problem of determining an optimal weighted coloring for a given set of intervals, (i.e., the off-line version), is NP-complete in the strong sense [9, page 226]. Recently, polynomial time approximation algorithms for the off-line version were obtained. Kierstead [10] showed that if the intervals are sorted by their weights, then the competitive ratio of First Fit is 80. Later, Kierstead [11] gave a different approximation algorithm which achieves a competitive factor of 6. Both algorithms use $\omega^*(G)$ as a benchmark for measuring the approximation factor.

The question of evaluating the performance of on-line algorithms was addressed by Sleator and Tarjan [19] who argued that the traditional approach of measuring the worst case behavior does not seem appropriate for many on-line algorithms. Therefore, they suggested a different measure, the *competitive ratio*. The performance of an on-line algorithm is compared with the performance of an *optimal* off-line algorithm that knows the sequence of events in advance. The maximum ratio between their respective performances, taken over all sequences, is called the competitive ratio. Extensive work has been done in recent years for finding the competitive ratio for different problems such as paging [19, 8], servers in a metric space [14], and managing a linked list [19]. We also adopt the competitive ratio as our performance measure. Notice that for dynamic storage allocation, it follows from Kierstead's work [11], that the competitive ratio (up to a constant factor) is the ratio between the area used by the on-line algorithm and $\omega^*(G)$.

1.1. Previous work. Dynamic storage allocation has been an important area of research since the 1950's [12, 18]. Coffman [3] surveyed the results and developments in this area until the early eighties. This area of research is also strongly linked with dynamic bin packing (See [4, 5]).

There are two natural heuristics that were developed early on for dynamic storage allocation: *First Fit* and *Best Fit*. First Fit finds the first free area that can fit the requirement of the current process; Best Fit finds the free area that best fits the requirement of the current process, i.e., the one that causes minimal fragmentation. A different category of methods for dynamic storage allocation constitute *segregated storage methods* [18]. These methods have many variations. The idea here is to partition the memory into blocks, such that in each block, only processes that have the same (or similar) requirement, are allocated.

Denote the maximum weight, taken over all intervals, by $W_{\max}$. In the early seventies, Robson [15, 16] gave an algorithm which had a competitive ratio of $O(\log(W_{\max}))$. He also showed that this is the best possible bound up to constant factors. Independently, Woodall [20] showed that the competitive ratio of First Fit is $O(\log(\omega *))$. Subsequently, Robson [17] showed that the competitive ratio of First Fit is also $O(\log(W_{\max}))$. It is interesting to note that Robson [17] also showed that the competitive ratio of Best Fit can be as bad as $W_{\max}$. As previously mentioned, algorithms for the off-line case with constant approximation factors have been recently obtained [10, 11].

1.2. Our contribution.

1.2.1. First Fit. Our first contribution is to provide tighter bounds for First Fit by considering a *new* parameter for analyzing the competitive ratio, i.e., the maximum number of concurrent active processes in memory.

Let the chromatic number of G be denoted by $\chi(G)$. Since interval graphs are perfect, $\chi(G) = \omega(G)$, where $\omega(G)$ denotes the clique number of G. Note that $\chi(G)$ is also the maximum number of concurrent active processes.

We show that First Fit achieves a competitive ratio of $\min\{O(\log(W_{\max})), O(\log(\chi(G)))\}$. This is an improvement over Robson [17], since, in general, $W_{\max}$ and $\chi(G)$ are incomparable. Tightening the analysis of First Fit is important, since it is a natural heuristic which is widely used.

It follows from the lower bounds proved by [15, 20] that our results are tight.

1.2.2. The Coloring algorithm. We next analyze a second deterministic online algorithm, which we call the *Coloring algorithm*. It belongs to the category of segregated storage methods [18], and uses an altogether different strategy than First Fit.

Consider a portion of storage area that is used by First Fit. Over time, we will generally see processes of diverse weights being allocated to that same portion. In fact, it is a fundamental property of First Fit to use “holes” created by processes that left the system, in order to allocate smaller weight processes. As discussed in the sequel, this property (i.e., allocating over time processes of various weights within the same area), is often a deficiency. Moreover, in an empirical study comparing different dynamic storage allocation methods, Detlefs et al. [6] show that hybrid algorithms that allocate differently small and large requirements, are very efficient in practice.

We now describe the Coloring algorithm. This algorithm never places processes of different weights (up to a constant factor of 2) within the same area. It assigns slots of storage area to each process “type” (according to weights); once a slot is assigned, the space within it is occupied only by processes of its type.

We show that the Coloring algorithm achieves the same (tight) worst-case competitive ratio as First Fit. To the best of our knowledge, the competitive factor of algorithms belonging to the category of segregated storage, has not been previously analyzed. In the following we briefly discuss several applications in which this type of algorithms can be expected to be efficient.

Consider the classical problem of allocating dynamic memory. It is well known that segregated storage algorithms are much more efficient than First Fit in finding free space for a new process. (See e.g., [18, 21]; see also [2] for an efficient implementation of First Fit). Indeed, by using standard techniques, the Coloring algorithm can allocate a new process within O(1) operations.

Dynamic storage allocation algorithms also arise in typical problems concerning storage of commodities. In such environments, one often needs to “prepare” a storage area that fits certain type of commodities. Since the weight of a commodity is usually related to its type, the Coloring algorithm saves the need to “reshape” storage areas for different types of commodities, whereas First Fit would frequently demand such actions.

Yet another field to which dynamic storage allocation algorithms are applicable is that of communication networks. Consider for example a radio network, in which transmission bandwidth is divided into time, or frequency, slots. Transmitting stations dynamically request variable numbers of consecutive slots from a network controller. After completing transmission, a station informs the controller on the release of its allocated bandwidth. Obviously, the controller should attempt to allocate bandwidth upon demand so as to minimize the total amount of used bandwidth. The application of dynamic storage allocation algorithms in such an environment is straightforward. Typically, there is a strong relation between the size of the requested bandwidth and the application that the corresponding transmission serves. This means that a receiving station, which attempts to follow the transmissions of a particular application, would need to be tuned to all portions of bandwidth in which such an application may be transmitted. First Fit would demand such a station to readjust its tuning frequently; the Coloring algorithm, on the other hand, guarantees that the receiving station would need just a few initial tunings.

1.2.3. Randomization. There has been extensive work devoted to exploiting and understanding the power of randomization in an on-line environment [1]. The idea is that randomized on-line algorithms might exhibit a better competitive ratio than deterministic ones, since they manage to "outsmart" the adversary sometimes.

A randomized on-line algorithm is defined to be a probability distribution over a space of deterministic on-line algorithms. There are several types of adversaries in this context, which differ by the information that the adversary is allowed to have. The most commonly considered adversary (and the one considered in this paper), is the *oblivious* one, who must construct the request sequence in advance based on the description of the algorithm, but has no access to the random choices made by the algorithm.

A randomized on-line algorithm A is said to have competitive ratio α if, for each request sequence, the ratio between the expected value of the performance of A and the off-line performance is at most α.

We show that, for the dynamic storage allocation problem, allowing randomized algorithms is not very helpful. In particular, we prove a lower bound of

$$\min\{O(\log W_{\max}/\log\log W_{\max}), O(\log\chi(G)/\log\log\chi(G))\}$$

on the competitive ratio of any randomized algorithm. Note that these lower bounds are only $\min\{O(\log\log W_{\max}), O(\log\log\chi(G))\}$ away from the deterministic lower and upper bounds. This is in contrast to the exponential gap between the competitive ratio of deterministic on-line algorithms to the competitive ratio for randomized on-line algorithms for the somewhat related paging problem: If k denotes the number of pages in the cache, then k is the competitive ratio of the best deterministic algorithm; on the other hand, it is known [8] that randomized algorithms can achieve a competitive ratio of $\log k$.

We conjecture that our randomized lower bounds can be further improved to match those of the deterministic case.

1.2.4. Multi storage devices. Our results can be extended to the case where there are several storage devices, and a process can be allocated to any of the devices. Simple analysis shows that having several devices has no (negative or positive) effect on the performance of dynamic storage allocation. The details can be found in [13].

2. Upper bounds. In this section we consider the First Fit and Coloring algorithms.

To clarify the exposition, we adopt the following notation. A process is referred to as an "item" where the item size is equal to the weight of the corresponding process. The storage device consists of cells, or locations, that are addressed consecutively from zero. The space which is used by an algorithm is equal to the value of the highest address of a location that was used by the algorithm.
We denote the chromatic number of G, $\chi(G)$, by k.

2.1. First Fit. First Fit is defined as follows. When a process appears, First Fit searches the memory for the first available area that satisfies the requirement of the process. In other words, the item is allocated in the area which has the lowest possible address.

We now bound the competitive ratio of First Fit. The idea is that locations with a "high" address will not contain "small" items. These notions are defined more rigorously in the following. Let $W = \lfloor W_{\max}/k \rfloor$.

LEMMA 2.1. *Items that have size less than or equal to* $4W$ *are always completely contained in the first* $5\omega^*$ *locations.*

Proof. Suppose First Fit needs to allocate area to an item I which has size $U \leq 4W$, and assume further that this cannot be done within the first $5\omega^*$ locations. In this case, the size of the largest hole in the first $5\omega^*$ locations is at most $U-1$. Since the number of holes is at most k, (including the area above the highest item), this means that the total area occupied by holes within the first $5\omega^*$ locations cannot exceed $k(U-1) < 4W_{\max} \leq 4\omega^*$. The area which is occupied by currently active items (of any size) cannot be more than $\omega^* - U$. Hence, there must be contiguous free space of size U within the first $5\omega^*$ locations, contradicting the assumption. ☐

For ease of presentation, we first bound the competitive ratio for the case where all weights are powers of two. Let ℓ be the smallest integer such that $2^\ell \geq 4W$. We show that in this case the following invariant holds.

> An item of size 2^j, where $j \geq \ell$, can always be allocated within the first $(6+j-\ell)\omega^*$ locations.

It follows from the invariant that the competitive ratio of First Fit is at most $5+\log_2 k$, by substituting $j = \log_2 W_{\max}$. (Note that j is an integer.)

The proof of the invariant is by induction on j. We first prove the base case where $j = \ell$. It follows from Lemma 2.1 that items that have size smaller than or equal to $4W$, are completely contained in the first $5\omega^*$ locations. Since we assumed that all item sizes are powers of 2, it follows that in locations with address higher than $5\omega^*$, the minimum item size is 2^ℓ. This implies that the minimum size of a hole above address $5\omega^*$, except perhaps for the first hole, is also 2^ℓ. Suppose that an item of size 2^ℓ cannot be allocated in the first $6\omega^*$ locations. This means that the only hole in locations with address between $5\omega^*+1$ and $6\omega^*$ is the first hole, which must have size less than 2^ℓ. This implies that the total size of the items that are currently active is strictly greater than ω^*, a contradiction. The inductive step is argued similarly, thus proving the invariant.

The analysis of First Fit in the general case is more subtle. Notice that it is not the case that one can round up the sizes to the closest power of two, and then argue that at most a factor of two in the competitive ratio is lost.

We first prove a technical lemma regarding harmonic numbers. Let $H_n = \displaystyle\sum_{i=1}^{n} \frac{1}{i}$.

LEMMA 2.2. *Assume that* $p \geq 1.5q$ *and* $q \geq 6$*. Then,* $H_p - H_q \geq 0.3$.

Proof. We use the following known bound on H_n [12, page 74]:

$$H_n = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \varepsilon$$

where γ is Euler's constant and $0 < \varepsilon < \frac{1}{252n^6}$. This implies that

$$H_p - H_q \geq \ln\left(\frac{p}{q}\right) + \frac{1}{2p} - \frac{1}{2q} + \frac{1}{12q^2} - \frac{1}{12p^2} + \frac{1}{120p^4} - \frac{1}{120q^4} - \frac{1}{252p^6}$$

Since $p \geq 1.5q$, we have that

$$\frac{1}{12q^2} - \frac{1}{12p^2} \geq \frac{1}{120q^4} - \frac{1}{120p^4} + \frac{1}{252p^6}$$

Also, since $q \geq 6$, we have that

$$\ln\left(\frac{p}{q}\right) + \frac{1}{2p} - \frac{1}{2q} > 0.3$$

Which concludes the proof. □

LEMMA 2.3. *The competitive ratio of First Fit is at most* $9 + 4\ln k$.

Proof. We claim that items of size at most j, where $j \geq W+1$, are always allocated by First Fit in locations with address lower than

$$5\omega^* + \frac{\omega^*}{0.3} \cdot \sum_{i=W+1}^{j} \frac{1}{i}$$

The claim is proved by induction. From Lemma 2.1, it follows that the claim holds for the case where $j \leq 4W$. We assume the claim holds for all values smaller than or equal to j (and at least $W+1$), and show that it holds for $j+1$. Let

$$L_i = 5\omega^* + \frac{\omega^*}{0.3} \cdot \sum_{\ell=W+1}^{i} \frac{1}{\ell}$$

Assume the claim does not hold for $j+1$. This implies that in the first L_{j+1} locations, there are no holes of size bigger than j. The following holds for locations between $(L_{i-1}+1)$ to L_i for $W+1 \leq i \leq j+1$, assuming that each item is completely contained in the area between $(L_{i-1}+1)$ and L_i, for some i:

> Any contiguous occupied area is of size at least i, and the holes are of size at most j. Hence, the total occupied area in locations between $(L_{i-1}+1)$ to L_i is at least
>
> $$\frac{i}{i+j} \cdot (L_i - L_{i-1})$$

Notice that, in the above, each hole is paired with the item below it. Thus, if an item is not completely contained in the area between $(L_{i-1}+1)$ and L_i, then the area it occupies within $(L_{i-2}+1)$ and L_{i-1} would not be not paired with any hole. This means that such an item can be paired with the first hole within $(L_{i-1}+1)$ and L_i, yielding that it is not necessary anymore to assume that each item is completely contained in the area between $(L_{i-1}+1)$ and L_i.

Summing up over all possible values of i, we get that the area that is occupied within the first L_{j+1} locations is at least:

$$\sum_{i=W+1}^{j+1} (L_i - L_{i-1}) \cdot \frac{i}{i+j} = \frac{\omega^*}{0.3} \cdot \sum_{i=W+1}^{j+1} \frac{1}{i} \cdot \frac{i}{i+j} = \frac{\omega^*}{0.3} \cdot \sum_{i=W+j+1}^{2j+1} \frac{1}{i}$$

Since $j \geq 4W$, it follows that $\frac{2j+1}{W+j+1} \geq 1.5$, and $W+j+1 \geq 6$. Hence, by Lemma 2.2,

$$\sum_{i=W+j+1}^{2j+1} \frac{1}{i} \geq 0.3$$

Implying that the area occupied within the first L_{j+1} locations is at least ω^*. This means that no new item can appear, contradicting the assumption that the above claim does not hold for $j+1$.

Hence, the total area occupied by First Fit is at most

$$5\omega^* + \frac{\omega^*}{0.3} \cdot \sum_{i=W+1}^{W_{\max}} \frac{1}{i}$$

Since $\ln n < H_n < \ln n + 1$, we get that the above is at most

$$5\omega^* + \frac{\omega^*}{0.3} \cdot \left(1 + \ln\left(\frac{W_{\max}}{W+1}\right)\right) \leq \omega^* \cdot \left(\frac{2.5 + \ln k}{0.3}\right) \leq \omega^* \cdot (9 + 4\ln k)$$

Thus proving the lemma. □

We thus have:

THEOREM 2.4. *The competitive ratio of First Fit is bounded by* $\min\{O(\log W_{\max}), O(\log k)\}$.

Proof. This follows from Lemma 2.3 together with Robson's proof [17]. □

Robson [15] and Woodall [20] showed a lower bound of $\Omega(\log W_{\max})$ on the competitive ratio of any deterministic on-line algorithm. We observe that, in the dynamic allocation problem instance they construct for the proof of the lower bound, $k = W_{\max}$. Hence, their lower bound extends immediately to a lower bound of $\min\{\Omega(\log W_{\max}), \Omega \log k\}$ on any deterministic algorithm. In fact, this observation can be generalized to hold in a more general setting, i.e., for $k < W_{\max}$, by slightly modifying the dynamic allocation problem instance in the proof of Woodall [20]. In this instance, items are inserted in iterations, where in the ith iteration, items of size 2^{i-1} are inserted. (Initially, $i = 1$.) The value of ω^* is fixed in advance by the adversary, and $W_{\max} = \omega^*$. A careful examination of Woodall's proof reveals that if the insertion process starts by inserting items of size 2^j, for any $j > 1$, then a lower bound of $\Omega(\log(\omega^*/2^j))$ on the competitive ratio can be proved. We note that, in this case, $k = O(\omega^*/2^j)$. Thus, a lower bound of $\Omega(\log k)$ on the competitive ratio of any deterministic on-line algorithm is proved for arbitrary $k < W_{\max}$. We conclude with the following corollary.

COROLLARY 2.5. *The competitive ratio of First Fit is* $\min\{\Theta(\log W_{\max}), \Theta(\log k)\}$.

2.2. The Coloring algorithm. In this section we analyze an algorithm for dynamic allocation, called the Coloring algorithm. The algorithm splits items into sets according to their sizes. An item of size l is assigned to set r, where $2^{r-1} < l \leq 2^r$. At any given time, each set r is assigned zero or more slots in memory, each of size 2^r. The assignment of slots is performed as follows. Suppose a new item belonging to the rth set appears. The Coloring algorithm tries to allocate it within a free slot assigned to that set. If no such slot is available, the algorithm produces a new slot by increasing the used area by 2^r. A slot assigned to set r (that is, the space allocated to it), is reserved forever for items belonging to that set. Thus, a location is never allocated with items belonging to different sets.

Let G_r denote the subgraph of G induced by intervals that have weight w, where $2^{r-1} < w \leq 2^r$. Denote by $\chi(G_r)$ the maximal number of concurrent items belonging to the rth set.

We observe that with respect to each set r, the Coloring algorithm uses at most twice the optimal amount of space by the following argument. Suppose that the size of each item in the rth set is 2^r. Since the intervals (corresponding to items) are sorted by their left endpoint, the greedy algorithm produces an optimal coloring. The factor of 2 is due to the fact that the size of each item in G_r is "rounded" up to the nearest power of 2. The following lemma seems to be folklore.

LEMMA 2.6. *The Coloring algorithm achieves a competitive ratio of* $O(\log(W_{\max}))$.

Proof. The amount of space assigned to a set r is precisely $\chi(G_r) \cdot 2^r$. For all values of r we have that $\chi(G_r) \cdot 2^r \leq 2 \cdot \omega^*$, meaning that each set occupies space of at most $2 \cdot \omega^*$ locations. The number of distinct sets is at most $\log(2W_{\max})$. Thus, the overall space used by the Coloring algorithm is at most $2 \cdot (\log(W_{\max}) + 1) \cdot \omega^*$. The lemma follows. □

LEMMA 2.7. *The total number of locations allocated to all sets r for which $2^r \leq \frac{W_{\max}}{k}$ is at most $2 \cdot W_{\max}$.*

Proof. As was observed in the previous proof, the total area assigned to a set r is precisely $\chi(G_r) \cdot 2^r$. Clearly, $\chi(G_r) \leq k$. This means that the total amount of space allocated to all sets for which $2^r \leq \frac{W_{\max}}{k}$ is at most

$$\sum_{r=0}^{\lfloor \log(\frac{W_{\max}}{k}) \rfloor} 2^r \cdot k \leq 2k \cdot \frac{W_{\max}}{k} = 2 \cdot W_{\max}$$

□

LEMMA 2.8. *The total amount of space allocated to all sets r for which $2^r \geq \lfloor \frac{W_{\max}}{k} \rfloor$ is at most $2\log(2k) \cdot \omega^*$.*

Proof. The number of distinct sets containing items of size larger than $\lfloor \frac{W_{\max}}{k} \rfloor$ is at most $\log(2k)$. Since each set occupies at most $2\omega^*$ locations, we get that the total area used is $2\log(2k) \cdot \omega^*$. □

THEOREM 2.9. *The competitive ratio of the Coloring algorithm is* $\min\{\Theta(\log W_{\max}), \Theta(\log k)\}$.

Proof. Follows directly from Lemmas 2.6, 2.7, 2.8, and the discussion preceding Corollary 2.5. □

2.3. Comparison. When comparing First Fit with the Coloring algorithm, it should be emphasized that neither algorithm dominates the other, i.e., for each algorithm there exist examples in which it outperforms the other. In particular, there are instances in which First Fit is near-optimal, whereas the Coloring algorithm reaches the worst-case bound.

The Coloring algorithm tends to perform well compared to First Fit in heterogeneous systems, i.e., when concurrency of items of various sizes is common. Weinstock and Wulf [21] describe an implementation of a segregated storage scheme called *Quick Fit*. It is interesting to note that they indicate that Quick Fit utilizes storage effectively in practice. The reason they provide for this phenomenon is that if a particular storage area has been allocated and then deallocated, there is a high probability that it will be allocated again.

3. A randomized lower bound. In this section we prove a lower bound of $O(\omega^* \log \omega^* / \log\log \omega^*)$ on the space required by any randomized on-line dynamic storage allocation algorithm. The lower bound will be proved for the case of a randomized oblivious adversary. (See Section 1.)

Our proof will use a corollary due to Yao [22] of von Neumann's minimax principle. Let $\mathcal{P}$ be a probability distribution defined on the input. Yao's result implies in our case that the expected value (with respect to $\mathcal{P}$) of the competitive ratio of any deterministic on-line algorithm is always a lower bound on the competitiveness of the best randomized on-line algorithm. Hence, our goal is to generate a "hard" distribution on the input, i.e., a distribution such that the expected value of the space required by any deterministic on-line algorithm is at least $O(\omega^* \log \omega^* / \log\log \omega^*)$.

Consider the following oblivious adversary *Adv*. *Adv* chooses a dynamic storage allocation instance probabilistically, and thus, *Adv* (implicitly) defines the probability distribution $\mathcal{P}$ on the input. *Adv* first decides on the value of ω^* before the run of the algorithm. The strategy of *Adv* is to insert into the system items of growing sizes, and then to delete half of them at random. *Adv* uses item sizes that are powers of g, where the value of g will be determined later. Without loss of generality it is assumed that ω^* is a power of 2, say $\omega^* = 2^r$. More specifically, *Adv* works in iterations, where in each it performs the following:

Insertion of items at iteration i: Denote by the random variable ω_i the total area occupied by items currently residing in the system (initially 0). *Adv* inserts $\left\lfloor \frac{\omega^* - \omega_i}{g^{i-1}} \right\rfloor$ items of size g^{i-1} into the system.

Deletion of items at iteration i: *Adv* deletes each item with probability 1/2, independently of the other items.

Starting with the the first iteration, *Adv* sequentially performs the insertion followed by the deletion steps for all iterations up to $i = \lfloor (r-1) \cdot \log_g 2 \rfloor + 1$.

Throughout the analysis, we mark at each iteration, following the insertion step, a subset of the items that currently reside in the system. Each such item will be termed as *marked*. A marked item may become unmarked at some later iteration. We denote by L_i the list of marked items, after the insertion step of iteration i, ordered according to their position in the storage device from bottom to top. The marking process is performed as follows.

- **Rule** A**:** At each iteration, all the new items inserted are marked.
- **Rule** B**:** At each iteration $i > 1$, we select approximately $1/(g+2)$ fraction of the marked items that resided in the system in the previous iteration (and were not deleted in the last deletion step). All other previously marked items become unmarked. The precise procedure is as follows. In the list L_{i-1}, we skip the first g items, and then search for the next item in the list that was not deleted in the deletion step of iteration $i-1$. This item is marked and inserted into the list L_i. The process is repeated until the list L_{i-1} is exhausted.

We stress that the notion of a marked item is solely for the purpose of analysis.

LEMMA 3.1. *Following the insertion step of the ith iteration, the expected number of marked items is at least*

$$\frac{i \cdot 2^{r-1}}{(g+2)^{i-1}} - 2$$

Proof. The claim clearly holds for $i = 1$. Assume inductively that it holds for iteration $i-1$, we show that it also holds for iteration i. By the above marking rules, marked items are either newly inserted items, or a subset of the items marked in the previous iteration.

The total area occupied by items, at any time, cannot be more than ω^*. Since each item is deleted with probability 1/2, the expected value of the occupied area, after the deletion step of any iteration, cannot be more than $\frac{\omega^*}{2} = 2^{r-1}$. This means that, for all i, the expected value of ω_i is at most 2^{r-1}. Hence, the expected number of newly inserted items at iteration i is at least

$$\frac{2^{r-1}}{g^{i-1}} - 1$$

and all these items are marked by Rule A.

We now evaluate the expected number of items marked by Rule B. For the purpose of the analysis, we assume that L_{i-1} is of infinite length; however, $|L_{i-1}|$ still refers to the original length of the list.

Let d_j be a random variable that counts the number of items in L_{i-1} between the $(j-1)$st marked item and the jth marked item. By Rule B and by the deletion strategy of Adv, $(d_j - g)$ is geometrically distributed with parameter $1/2$. This means that $\text{Exp}[d_j] = g + 2$. Let T be a random variable that denotes the smallest t such that

$$\sum_{j=1}^{t} d_j > |L_{i-1}|$$

It is clear that the expected number of items marked by Rule B is $\text{Exp}[T] - 1$. Notice that a stopping rule can be associated with Rule B in a natural way: stop whenever the list L_{i-1} is exhausted. Hence, Wald's identity [7] can be applied in this case; it states that,

$$\text{Exp}[T] \cdot \text{Exp}[d_i] = \text{Exp}\left[\sum_{j=1}^{T} d_j\right].$$

Clearly,

$$|L_{i-1}| < \text{Exp}\left[\sum_{i=1}^{T} d_i\right] \leq |L_{i-1}| + g + 2.$$

Therefore, we get that the expected number of items marked by Rule B is at least

$$\frac{|L_{i-1}|}{g+2} - 1.$$

Let ℓ_i be a random variable that denotes the cardinality of L_i. From the above it follows that

$$\text{Exp}[\ell_i | \ell_{i-1}] \geq \frac{\ell_{i-1}}{g+2} - 1 + \frac{2^{r-1}}{g^{i-1}} - 1.$$

Since $\text{Exp}[\ell_i | \ell_{i-1}]$ is a linear function of ℓ_{i-1}, it follows that

$$\text{Exp}[\ell_i] \geq \frac{\text{Exp}[\ell_{i-1}]}{g+2} - 1 + \frac{2^{r-1}}{g^{i-1}} - 1.$$

By the inductive assumption,

$$\text{Exp}[\ell_{i-1}] \ \geq \ \frac{(i-1) \cdot 2^{r-1}}{(g+2)^{i-2}} - 2.$$

Thus, the expected number of items marked by rules A and B is at least

$$\begin{aligned}
\operatorname{Exp}[\ell_i] &\geq \left(\frac{(i-1)\cdot 2^{r-1}}{(g+2)^{i-2}} - 2\right)\cdot\frac{1}{g+2} - 2 + \frac{2^{r-1}}{g^{i-1}} \\
&\geq \frac{i\cdot 2^{r-1}}{(g+2)^{i-1}} - 2 + \frac{2^{r-1}((g+2)^{i-1}-g^{i-1})}{g^{i-1}(g+2)^{i-1}} - \frac{2}{g+2} \\
&\geq \frac{i\cdot 2^{r-1}}{(g+2)^{i-1}} - 2 + \frac{(g+2)^{i-1}-g^{i-1}}{(g+2)^{i-1}} - \frac{2}{g+2} \\
&= \frac{i\cdot 2^{r-1}}{(g+2)^{i-1}} - 2 + \frac{1}{g+2}\left(g+2-\left(\frac{g}{g+2}\right)^{i-2}\cdot g - 2\right) \\
&\geq \frac{i\cdot 2^{r-1}}{(g+2)^{i-1}} - 2. \qquad (1)
\end{aligned}$$

□

We define the distance between two items residing in the storage device as the distance between their centers.

LEMMA 3.2. *Following the insertion step of the ith iteration, the distance between two consecutive marked items is at least* $g^{i-1}/2$.

Proof. We prove the lemma by induction. The claim trivially holds for the first iteration. Assuming that it holds for iteration $i-1$, we prove it for iteration i. Let X and Y be two consecutive items in L_i. If one of them was marked by Rule A then its size is g^{i-1}, thus the distance between the two centers is at least $g^{i-1}/2$. Suppose, then, that both items were marked by Rule B. This means that both X and Y also belong to L_{i-1}. Recall that Rule B skips at least g items in L_{i-1} between X and Y. Coupling this with the inductive hypothesis, we conclude that the distance between X and Y in iteration i is at least

$$g\cdot\frac{g^{i-2}}{2} = \frac{g^{i-1}}{2}.$$

□

THEOREM 3.3. *The competitive ratio of any randomized dynamic storage allocation algorithm is at least* $\Omega(\log\omega^*/\log\log\omega^*)$.

Proof. Consider the storage device after the insertion step of iteration i. By Lemmas 3.1 and 3.2, the occupied storage area is at least

$$\left(\frac{i\cdot 2^{r-1}}{(g+2)^{i-1}} - 2\right)\cdot\frac{g^{i-1}}{2} = \frac{i\cdot 2^{r-2}}{2\cdot\left(1+\frac{2}{g}\right)^{i-1}} - g^{i-1} \geq \frac{i\cdot 2^{r-2}}{2\cdot\left(1+\frac{2}{g}\right)^{i-1}} - 2^{r-1}$$

where the last transition follows from $i \leq \lfloor (r-1)\cdot\log_g 2\rfloor + 1$. Hence, after the insertion step of the last iteration, the occupied storage area is at least

$$\frac{\left((r-1)\cdot\log_g 2\right)\cdot 2^{r-2}}{2\cdot\left(1+\frac{2}{g}\right)^{(r-1)\log_g 2}} - 2^{r-1}.$$

We choose g such that $g = \lfloor r/\log_2 r\rfloor$. This means that the occupied storage area is at least

$$\frac{(r-1)\cdot 2^{r-2}\cdot \frac{1}{\log_2 r - \log_2\log_2 r}}{2\cdot\left(1+\frac{2}{\frac{r}{2\log_2 r}}\right)^{\frac{2r}{\log_2 r}}} - 2^{r-1} \;\geq\; \frac{\frac{r-1}{\log_2 r}\cdot 2^{r-2}}{2\cdot e^8} - 2^{r-1} \;=\; \Omega\left(\frac{\omega^*\cdot\log\omega^*}{\log\log\omega^*}\right).$$

□

Since $\omega^* \geq W_{\max}$ and $\omega^* \geq k$, it turns out that $\Omega(\log W_{\max}/\log\log W_{\max})$ and $\Omega(\log k/\log\log k)$ are also lower bounds on the competitive ratio. Note that in the strategy of Adv the three values ω^*, $W_{\max}$ and k are equal up to a constant factor.

Acknowledgements. We thank the two anonymous referees for clarifying the presentation of the paper. Many thanks to Oran Sharon for bringing this problem to our attention. We would like to thank Yishay Mansour and Ron Shamir for helpful discussions.

REFERENCES

[1] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Wigderson, *On the power of randomization in on-line algorithms*, Algorithmica, 11, (1994), pp. 2-14.

[2] R. P. Brent, *Efficient implementation of the first-fit strategy for dynamic storage allocation*, ACM Trans. on Programming Languages and Systems, 11, (1989), pp. 388-403.

[3] E. G. Coffman, *An introduction to combinatorial models of dynamic storage allocation*, SIAM Review, 25, (1983), pp. 311-325.

[4] E. G. Coffman, M. R. Garey and D. S. Johnson, *Dynamic bin packing*, SIAM J. on Comput., 12, (1983), pp. 227-258.

[5] E. G. Coffman, M. R. Garey and D. S. Johnson, *Approximation algorithms for bin-packing - an updated survey*, In: Algorithm design for computer system design, pp. 49-106, edited by G. Ausiello, M. Lucertini and P. Serafini, Springer Verlag, Wien - New York, 1984.

[6] D. Detlefs, A. Dosser and B. Zorn, *Memory allocation costs in large C and C++ programs*, Technical Report CU-CS-665-93, University of Colorado, Boulder, (1993).

[7] W. Feller, *An introduction to the theory of probability and its applications*, John Wiley and Sons, (1967).

[8] A. Fiat, R. M. Karp, M. G. Luby, L. A. McGeoch, D. D. Sleator and N. E. Young, *Competitive paging algorithms*, J. of Algorithms, 12, (1991), pp. 685-699.

[9] M. R. Garey and D. S. Johnson, *Computers and intractability - a guide to the theory of NP-completeness*, W. H. Freeman, San Francisco, (1979).

[10] H. A. Kierstead, *The linearity of first-fit coloring of interval graphs*, SIAM J. Disc. Math, 1, (1988), pp. 526-530.

[11] H. A. Kierstead, *A polynomial time approximation algorithm for dynamic storage allocation*, Discrete Mathematics, 88, (1991), pp. 231-237.

[12] D. E. Knuth, *Fundamental algorithms, Vol. 1*, 2nd Edition, Section 2.5, Addison-Wesley, Reading, MA, (1973).

[13] M. G. Luby, J. Naor and A. Orda, *Tight Bounds for Dynamic Storage Allocation*, Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, (1994), pp. 724-732.

[14] M. S. Manasse, L. A. McGeoch and D. D. Sleator, *Competitive algorithms for on-line problems*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, (1988), pp. 322-333.

[15] J. M. Robson, *An estimate of the store size necessary for dynamic storage allocation*, J. of the ACM, 18, (1971), pp. 416-423.

[16] J. M. Robson, *Bounds for some functions concerning dynamic storage allocation*, J. of the ACM, 12, (1974), pp. 491-499.

[17] J. M. Robson, *Worst case fragmentation of first fit and best fit storage allocation strategies*, Computer Journal, 20, (1977), pp. 242-244.

[18] T. A. Standish, *Data structures techniques*, Addison-Wesley Publishing Company, (1980).

[19] D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, C. of the ACM, 28, (1985), pp. 202-208.

[20] D. R. Woodall, *The bay restaurant - a linear storage problem*, American Mathematical Monthly, 81, (1974), pp. 240-246.

[21] C. B. Weinstock and W. A. Wulf, *Quickfit: an efficient algorithm for heap storage allocation*, ACM SIGPLAN Notices, 23, (1988), pp. 141-144.

[22] A. C-C. Yao, *Probabilistic computations: towards a unified measure of complexity*, Proceedings of the 17th Annual Symposium on Foundations of Computer Science, (1977), pp. 222-227.