Computing 49, 329–338 (1993)

Computing
© Springer-Verlag 1993
Printed in Austria

Repacking Helps in Bounded Space On-Line Bin-Packing*

G. Galambos, Szeged and **G. J. Woeginger**, Graz

Received August 20, 1991

Abstract — Zusammenfassung

Repacking Helps in Bounded Space On-Line Bin-Packing. We consider a version of the on-line bounded-space bin-packing problem where repacking the items within the active bins is allowed. For this problem, the 1.69103 lower bound of Lee and Lee [7] for the worst case ratios of bounded-space approximation algorithms still applies. We present a polynomial time approximation algorithm that reaches the best possible worst case ratio matching the Lee and Lee lower bound while using only *three* active bins.

AMS Subject Classifications: 90B35, 90C27.

Key words: Combinatorial problems, on-line, bin packing, suboptimal algorithms.

Beim Bounded Space On-line Bin-Packing kann Umpacken von Vorteil sein. Wir behandeln eine Variante des On-Line Bound-Space Bin-Packings, in der das Umpacken der Gegenstände innerhalb der aktiven Bins erlaubt ist. Auch für diese Variante gilt die untere Schranke 1.69103, die Lee und Lee [7] für Worst Case Ratios von Bounded-Space Approximations-Algorithmen bewiesen haben. Wir konstruieren einen bestmöglichen polynomialen Approximations-Algorithmus, der die Schranke von Lee und Lee erreicht und dazu nur *drei* aktive Bins verwendet.

1. Introduction

In the classical one-dimensional bin-packing problem, we are given a list of items $L = (a_1, a_2, \ldots, a_n)$, each item $a_i \in (0, 1]$, and we must find a packing of these items into a minimum number of unit-capacity bins. This problem arises in a wide variety of contexts and has been studied extensively since the early 1970s (see e.g. [5] and [6]). Since the problem of finding an optimal packing is NP-hard, research has concentrated on approximation algorithms that find near-optimal packings and use polynomial time. Let $\mathrm{OPT}(L)$ and $A(L)$ denote, respectively, the number of bins used by an optimum algorithm and the number of bins used by a heuristic algorithm A to pack the input list L. Then the *worst case ratio* or *performance guarantee* of A, denoted by $r(A)$ is defined as

$$\lim_{Opt(L)\to\infty} \sup_{L} A(L)/\mathrm{OPT}(L).$$

This ratio is customarily used to measure the quality of a heuristic bin-packing algorithm.

* This paper was supported by a grant from the Hungarian Academy of Sciences (OTKA Nr. 2037).

A bin-packing algorithm is called *on-line* if it packs all items a_i solely on the basis of the sizes of the items a_j, $1 \leq j \leq i$ and without any information on subsequent items. A bin-packing algorithm uses *k-bounded space* if for each item a_i, the choice of bins to pack it into is restricted to a set of k or fewer *active* bins, where each bin becomes active when it receives its first item, but once it is declared inactive (or *closed*), it can never become active again.

These latter restrictions (on-line and bounded-space) arise in many applications, as in packing trucks at a loading dock or in communicating via channels with bounded buffer size. Consequently, the problem was analyzed thoroughly in the 1980s, cf. the papers by Csirik and Imreh [1], Csirik and Johnson [2], Lee and Lee [7] and Woeginger [8]. On the negative side, Lee and Lee [7] proved that no on-line approximation algorithm using bounded-space can have a performance guarantee less than the constant $h_\infty \approx 1.69103$. On the positive side, algorithms were detected whose asymptotic worst-case ratios approach h_∞ as the number of active bins tends to infinity; but there is no algorithm known that reaches this bound while using only a finite number of bins.

In this paper we will consider a related bounded-space bin-packing problem where *repacking the k active bins* is allowed. In this version of the problem, we are allowed to perform the standard actions of bounded-space bin-packing, i.e. we are allowed to

(i) Open a new bin (if the number of active bins is less or equal to $k - 1$),
(ii) Close some active bin (and never open it again),
(iii) Pack a new item into some active bin (if the contents of the bin remains below one).

But in contrary to standard bin-packing, we are also allowed to (let $\mathscr{B}_1, \ldots, \mathscr{B}_k$ denote the active bins; we will identify the contents of a bin with the bin)

(iv) Repack the set of active bins, i.e. to form a new partition $\mathscr{B}'_1, \ldots, \mathscr{B}'_k$ of the items inside the active bins such that $\bigcup \mathscr{B}_i = \bigcup \mathscr{B}'_i$ holds, and such that the items in each part of the new partition have overall size less or equal to one.

To allow action (iv) is a natural assumption. As long as an item is in some active bin, the item is available for the "packer" and the packer may change its position. In the loading dock example mentioned above, trucks will partially be repacked and items will be moved from one truck at the loading dock to another in order to increase the number of pieces packed.

Unfortunately, the h_∞ lower bound of Lee and Lee carries over to this problem, too. But we will show that now there exist approximation algorithms using a *finite* number of active bins and reaching this worst case ratio. More precisely, we will present an algorithm called REP$_3$ that uses only *three* active bins and has performance guarantee h_∞. The running time of REP$_3$ is $O(n^2)$. The main tool we apply is a new weighting function function first used in [8].

The paper is organized as follows. Section 2 gives some basic definitions and introduces the weighting function. In Section 3 we formulate the algorithm REP$_3$ and analyze its worst case performance. Section 4 finishes with the discussion.

2. Definitions and Preliminaries

The following sequence that was introduced by Golomb [4] will be essential in the definition and in the analysis of our algorithm REP$_3$.

$$t_1 = 2$$
$$t_{i+1} = t_i(t_i - 1) + 1 \qquad \text{for } i \geq 1$$

Knowing this, the exact definition of the real number h_∞ (that was already used in the introduction) may be stated as

$$h_\infty = \sum_{i=1}^{\infty} \frac{1}{t_i - 1} = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \frac{1}{1806} + \cdots \approx 1.69103.$$

The proof of the following theorem is an easy modification of the proof of Theorem 3 in [7].

Theorem 1. *Let A be any bounded-space on-line bin-packing algorithm that uses repacking. Then $r(A) \geq h_\infty$ holds.*

Proof. Let n and m be some integers and let ε be a very small positive real. We consider the list $L(n, m)$ that consists of m homogeneous sublists $L_i(n)$, $1 \leq i \leq m$, each of length n. The sublist $L_i(n)$ contains n items of size $1/t_i + \varepsilon$.

Obviously, the optimum packing of $L(n, m)$ uses exactly n bins where every bin contains exactly one item from every sublist. On the other hand, any k-bounded space algorithm must use n different bins to pack the first sublist $L_1(n)$. At most k of these n bins are active, when the first item of sublist $L_2(n)$ arrives; consequently, the algorithm requires at least $n/2 - k$ new bins to pack sublist $L_2(n)$. Using analogous arguments we conclude that any heuristic will use at least $n\sum_{i=1}^{m} 1/(t_i - 1) - (m - 1)k$ bins. Since k is constant, the quotient of this expression and OPT$(L(n, m))$ comes arbitrarily close to h_∞ as n and m both tend to infinity. □

Next, let us define our weighting function $W(x)$: $(0, 1] \to R^+$.

$$W(x) = \begin{cases} x + \dfrac{1}{t_{i+1} - 1} & \text{for } 1/t_i < x \leq 1/(t_i - 1) \text{ and } 1 \leq i \\[2ex] \dfrac{t_i + 1}{t_i} \cdot x & \text{for } 1/(t_{i+1} - 1) < x \leq 1/t_i \text{ and } 1 \leq i \end{cases}$$

An illustration for the weight of items with size close to one is given in Table 1. The weight of a set of items is defined to be the sum of weights over all items in the set. The weight of a bin is the weight of all items in it. Analogously, we define the size of a set of items (of a bin) to be the sum of sizes over all items in the set (in the bin). The proofs of the following observations are straightforward.

Observation 2

(i) *$W(x)$ is nondecreasing in* $(0, 1]$. □
(ii) *For $i \geq 1$ and $x \leq 1/t_i$, $W(x)/x \leq (t_i + 1)/t_i$ holds.* □
(iii) *For $i \geq 1$ and $x \geq 1/(t_{i+1} - 1)$, $W(x)/x \geq (t_i + 1)/t_i$ holds.* □

Table 1. Illustration for the weighting function W

Type	Interval	Weight(x)	Type	Interval	Weight(x)
A	$(1/2, 1]$	$x + 1/2$	B_3	$(1/7, 1/6]$	$x + 1/42$
B_2	$(1/3, 1/2]$	$x + 1/6$	C_3	$(1/8, 1/7]$	$8x/7$
C_2	$(1/4, 1/3]$	$4x/3$	D_3	$(1/42, 1/8]$	$8x/7$
D_2	$(1/6, 1/4]$	$4x/3$	B_4	$(1/43, 1/42]$	$x + 1/1806$

Lemma 3. *In any packing of a list L the weight of any bin is at most h_∞. Hence, $W(L) \le h_\infty \,\mathrm{OPT}(L)$ holds.*

Proof. Let us consider some fixed bin $\mathscr{B}$ that contains items $q_1 \ge q_2 \ge \cdots \ge q_m$. We distinguish two cases.

(a) $q_i \in (1/t_i, 1/(t_i - 1)]$ for $i = 1 \ldots m$. Then

$$W(\mathscr{B}) = \sum_{i=1}^{m} \left(q_i + \frac{1}{t_{i+1} - 1} \right) = \sum_{i=1}^{m} q_i + \sum_{i=2}^{m} \frac{1}{t_i - 1}$$
$$\le 1 + \sum_{i=2}^{m} \frac{1}{t_i - 1} < h_\infty .$$

(b) Now assume $r \le m$ is the least i such that $q_r \le 1/t_r$. We denote by Q the sum $\sum_{i=r}^{m} q_i < 1/(t_r - 1)$. By Observation 2.ii, the total weight of $q_r \ldots q_m$ is less or equal to $(t_r + 1)Q/t_r$. This yields

$$W(\mathscr{B}) \le 1 - Q + \sum_{i=1}^{r-1} \frac{1}{t_{i+1} - 1} + \frac{t_r + 1}{t_r} \cdot Q$$
$$< \sum_{i=1}^{r} \frac{1}{t_i - 1} + \frac{1}{t_r(t_r - 1)} < h_\infty$$ □

3. The Repacking Algorithm

In this section we define and analyze the *Repacking Algorithm*, REP$_3$ for short. This algorithm always keeps three active bins that we call BIN$_1$, BIN$_2$ and BIN$_3$. REP$_3$ proceeds as follows.

```
(1) Get a new item x and put x into an empty active bin.
(2) Repack the three active bins such that at least one bin is empty
    or such that at least one bin has weight greater or equal to one.
(3) Close all active bins with weight at least one and open new bins
    instead of them. Goto (1).
```

Of course the crucial (and not well-defined) step of the algorithm REP$_3$ is Step (2). The main part of this section is devoted to establishing the fact that Step (2) is always possible and how to perform it. To do this, we first prove the following theorem.

Theorem 4. *Let* BIN$_1$, BIN$_2$, BIN$_3$ *be three bins. Then we can either repack the bins to produce a packing with an empty bin or we can find a subset of items with weight at least one and size at most one.*

Proof. We will assume that it is neither possible to produce a *good packing* (a packing with an empty bin) nor to find a *good subset* of items (a subset of weight greater or equal one and of size at most one), and we will derive a contradiction from this. It is convenient to classify the items according to the following partition of the unit-interval (cf. Table 1).

$$\begin{aligned} A &= (1/2, 1] & & \\ B_i &= (1/t_i, 1/(t_i - 1)] & & \text{for } i \geq 2 \\ C_i &= (1/(t_i + 1), 1/t_i] & & \text{for } i \geq 2 \\ D_i &= (1/(t_{i+1} - 1), 1/(t_i + 1)] & & \text{for } i \geq 2 \end{aligned}$$

First of all, we will produce a *First-Fit Decreasing packing* (FFD-packing for short) of the items in the three active bins (cf. [5]). That means that we first order the items by decreasing size; then we go through the sorted list and place each successive piece into the first (leftmost) active bin into which it will fit. Since we assumed that there neither exists a good packing nor a good subset, in the produced FFD-packing neither BIN$_3$ will be empty nor will there be a bin with weight greater or equal one.

We will proof a number of combinatorial properties of the FFD-packing. The proof is split into several claims.

Claim 1. *In the* FFD-*packing, no bin contains an A-item x.*

Proof. Trivial, as every type-A item has weight at least one and would form a good subset. □

Claim 2. *In the* FFD-*packing, neither* BIN$_2$ *nor* BIN$_3$ *contains any D_i-item x, $i \geq 2$.*

Proof. Since x was not put into BIN$_1$, BIN$_1$ is at least $t_i/(t_i + 1)$ full. By Observation 2.iii, for $x \geq 1/(t_{i+1} - 1)$, $W(x)/x \geq (t_i + 1)/t_i$ must hold. But now the total weight of BIN$_1$ is at least

$$\frac{t_i}{t_i + 1} \cdot \frac{t_i + 1}{t_i} = 1,$$

and the contents of BIN$_1$ would be a good subset. □

Claim 3. *In the* FFD-*packing, neither* BIN$_2$ *nor* BIN$_3$ *contains any B_i-item x, $i \neq 3$. Moreover,* BIN$_2$ *and* BIN$_3$ *together contain at most one B_3-item.*

Proof. First we show that BIN$_2$ and BIN$_3$ cannot contain a B_2-item. By Claim 1, BIN$_1$ does not contain an A-item. If BIN$_2$ or BIN$_3$ contains a B_2-item, BIN$_1$ must contain two B_2-items, and consequently, Bin_1 has weight at least $2(1/3 + 1/6) = 1$. Again the contents of Bin_1 would form a good subset.

Next, we examine the case $i \geq 3$. We denote by X the overall size of all items in Bin_1 that are larger than the B_i-items. By Observation 2.iii, the total weight of these items is at least $(t_{i-1} + 1)X/t_{i-1}$. Let the number of B_i-items in Bin_1 be b, let their overall size be denoted by B. We use the fact that $W(\text{Bin}_1) < 1$ must hold and get

$$\frac{t_{i-1} + 1}{t_{i-1}} X + B + \frac{b}{t_{i+1} - 1} < 1. \tag{1}$$

As the item $x \leq 1/(t_i - 1)$ did not fit into Bin_1, we know that

$$X + B > \frac{t_i - 2}{t_i - 1} \tag{2}$$

must hold. We subtract the equation (2) multiplied by $(t_{i-1} + 1)$ from (1) multiplied by t_{i-1} and derive

$$\frac{t_{i-1}}{t_{i+1} - 1} b < t_{i-1} - \frac{t_i - 2}{t_i - 1}(t_{i-1} + 1) + B. \tag{3}$$

On the other hand, we know that every B_i-item in Bin_1 has size at most $1/(t_i - 1)$ and that there are exactly b such items. This implies $B \leq b/(t_i - 1)$. Plugging this into equation (3) and simplifying the resulting inequality yields

$$b > \frac{t_i - t_{i-1} - 2}{t_i - t_{i-1}} \cdot t_i = t_i - \frac{2t_i}{t_i - t_{i-1}}. \tag{4}$$

Now for $i \geq 4$, the righthand side of (4) is at least $t_i - 3$, and therefore, b is at least $t_i - 2$. Together with the item x in Bin_2, there are at least $t_i - 1$ B_i-items, each of size greater $1/t_i$. Hence, the total weight of these $t_i - 1$ B_i-items is at least

$$(t_i - 1) \cdot W\left(\frac{1}{t_i} + \varepsilon\right) > (t_i - 1) \cdot \left(\frac{1}{t_i} + \frac{1}{t_{i+1} - 1}\right) = 1,$$

and it is easy to check that they fit into a single bin. Thus, we have constructed a good subset and derived a contradiction.

For $i = 3$, the righthand side of (4) is 7/2, and this yields $b \geq 4$. If there are two or more B_3-items in Bin_2 and Bin_3, we have at least six B_3-items and we can argue analogously to above. □

Claim 4. *In the* FFD*-packing, neither* Bin_2 *nor* Bin_3 *contains any* C_2*-item* x*. The bin* Bin_3 *does not contain any* B_3*-item* y*.*

Proof. Assume the contrapositive. As x did not fit into Bin_1, Bin_1 must contain three C_2-items, each of size at least 1/4. This gives for Bin_1 an overall weight of at least one, and the first part of the claim is proven.

To see the correctness of the second part, we observe that Bin_2 and Bin_3 do not contain A-, B_2-, C_2- or D_2-items. Consequently, the B_3-item y is the largest item in these two bins, and FFD puts it into Bin_2. □

Claim 5. *In the* FFD*-packing* BIN$_3$ *cannot contain any* C_i*-item* x *with* $i \geq 4$.

Proof. We denote by Y the overall size of all items in BIN$_2$ that are larger than the C_i-items. All of these items are at least C_{i-1}-items, and by Observation 2.iii, the total weight of them is at least $(t_{i-1}+1)Y/t_{i-1}$. Let the number of C_i-items in BIN$_2$ be c, let their overall size be denoted by C. Since the contents of BIN$_2$ is not a good subset, $W(\text{BIN}_2) < 1$ holds and this yields

$$\frac{t_{i-1}+1}{t_{i-1}}Y + \frac{t_i+1}{t_i}C < 1. \tag{5}$$

Since the item $x \leq 1/t_i$ did not fit into BIN$_2$, we have

$$Y + C > \frac{t_i - 1}{t_i}. \tag{6}$$

We multiply the inequality (6) by $(t_{i-1}+1)$ and the inequality (5) by the factor t_{i-1}. Subtracting one inequality from the other and simplifying gives

$$C > \frac{t_i - t_{i-1} - 1}{t_i - t_{i-1}} = 1 - \frac{1}{t_i - t_{i-1}}. \tag{7}$$

Moreover, $C \leq c/t_i$ holds. Combining this with inequality (7), we derive

$$c > t_i - \frac{t_i}{t_i - t_{i-1}} > t_i - 2, \tag{8}$$

where the righthand inequality in (8) follows from $i \geq 4$. Summarizing, we have at least $(t_i - 1)$ C_i-items in BIN$_2$ and at least one C_i-item in BIN$_3$. This gives t_i items with overall weight at least one and overall size at most one, and we can construct a good subset. □

Claim 6. *In the* FFD*-packing* BIN$_3$ *cannot contain any* C_3*-item* x.

Proof. If BIN$_2$ does not contain a B_3-item, it must contain seven C_3-items with overall weight at least one (otherwise FFD puts x into BIN$_2$, too). Hence, we may assume that BIN$_2$ does contain some B_3-item y.

Similarly as in the proof of Claim 3, we denote by X the overall size of all items in BIN$_1$ that are larger than the B_3-items (i.e. larger than 1/6). By Observation 2.iii, the total weight of these items is at least $4X/3$. Let the number of B_3-items in BIN$_1$ be b, and let their overall size be denoted by B. Then $W(\text{BIN}_1) < 1$ implies $4X/3 + B + b/42 < 1$. As the C_3-item $x \leq 1/7$ did not fit into BIN$_1$, we have $X + B > 6/7$. These two inequalities yield $14B > b + 6$. Finally, we plug in $B \leq b/6$ and derive $b > 9/2$. Altogether, there are at least six B_3-items, and once more we detected a good set of items with size at most one and weight at least one. □

Summarizing, in Claims 1 through 6 we have shown that in the FFD-packing BIN$_3$ can neither contain an A-, nor a B_i-, nor a C_i-, nor a D_i-item. Consequently, BIN$_3$ is empty and the FFD-packing is a good packing. This is the final contradiction and the proof of Theorem 4 is complete. □

Theorem 5. *The algorithm* REP$_3$ *can be implemented in such a way that it never gets stuck in Step* (2). *It has the best possible worst case ratio* h_∞, *and a running time of* $O(n^2)$ (*where n denotes the number of items*).

Proof. The proof that REP$_3$ can be implemented such that it never gets stuck is done by induction over the number of packed items. We will keep the followng three invariants. As REP$_3$ receives a new item to pack, (i) BIN$_3$ is empty, (ii) BIN$_1 \cup$ BIN$_2$ does not contain any A-item and (iii) BIN$_1 \cup$ BIN$_2$ contains at most one B_2-item.

Obviously, the three invariants hold as REP$_3$ receives the first item. Assume they hold after the packing of item a_k, and consider the moment as REP$_3$ receives a_{k+1}. In Step (1), item a_{k+1} is put into BIN$_3$. If a_{k+1} is an A-item, we just move on to Step (3) where BIN$_3$ is closed and replaced by an empty bin. Obviously, all three invariants hold again. Otherwise, a_{k+1} has size at most 1/2 and we distinguish three subcases.

(a) If a_{k+1} is a B_2-item and BIN$_1 \cup$ BIN$_2$ contains some B_2-item z, we simply move z into BIN$_3$ and go to Step (3).

(b) BIN$_1 \cup$ BIN$_2$ does not contain any B_2-item. In this case, we apply Theorem 4. Either we can repack all items into two bins such that BIN$_3$ is empty (in this case we go to Step (3) and all three invariants are fulfilled), or we detect a good subset $\mathscr{S}$. All items in $\mathscr{S}$ are of size at most 1/3, and from Observation 2.iii we derive that the total size of $\mathscr{S}$ is at least 3/4.

    We pack all items of $\mathscr{S}$ into BIN$_3$, a_{k+1} moves into BIN$_1$. The remaining pieces are packed by FFD into BIN$_1$ and BIN$_2$. We claim that all pieces can be packed. Assume that some piece z does not fit in any bin. If $z > 1/4$, at least one larger piece has been packed into BIN$_1$, and the contents of BIN$_1$ is at least $a_{k+1} + 1/4$. If $z \le 1/4$, the contents of BIN$_1$ is at least $3/4 \ge a_{k+1} + 1/4$. Thus, in any case the contents of BIN$_1$ is at least $a_{k+1} + 1/4$. But now we have derived a contradiction: The overall size of all pieces is at most $2 + a_{k+1}$. BIN$_3$ contains at least 3/4, BIN$_1$ contains at least $a_{k+1} + 1/4$, and all the remaining pieces fit into BIN$_2$. Consequently, we may go on to Step (3).

(c) If $a_{k+1} \le 1/3$ and BIN$_1 \cup$ BIN$_2$ contains some B_2-item of size greater 1/3, we proceed similarly to subcase (b). We apply Theorem 4. If we can repack all items into two bins such that BIN$_3$ is empty, everything is fine. Hence, assume we detect a good subset $\mathscr{S}$. All items in $\mathscr{S}$ are of size at most 1/2, and from Observation 2.iii we derive that the total size of $\mathscr{S}$ is at least 2/3.

    Again we pack all items of $\mathscr{S}$ into BIN$_3$, a_{k+1} into BIN$_1$ and apply FFD to the remaining pieces. If some piece z could not be packed, it is easy to see that $z \le 1/3$ and the contents of BIN$_1$ must be at least $2/3 \ge 1/3 + a_{k+1}$. We end with a contradiction as in subcase (b).

This completes the inductional proof.

To prove that REP$_3$ has the best possible worst case ratio h_∞, we simply observe that for any list L of items, REP$_3(L) \le W(L) + 3$ holds (The algorithm closes only bins of weight at least one, the last three active bin are added to this number). Combining this with the inequality in Lemma 3, we get

$$\text{REP}_3(L) - 3 \leq W(L) \leq h_\infty \, \text{OPT}(L).$$

Together with the lower bound of Theorem 1 this implies $r(\text{REP}_3) = h_\infty$.

Finally, we consider the time complexity of REP_3. If we store all elements in the active bins in a sorted binary tree, inserting and removing some item can be done in $O(\log n)$ time. Hence, all the sorting steps can be performed in overall time $O(n \log n)$. Every time a new item arrives, we have to repack at most n items. This gives an overall time complexity of $O(n^2)$. □

4. Discussion

In this paper we introduced an on-line bin-packing algorithm REP_3 that uses only bounded space and strongly exploits the possibility of *repacking*. In contrary to all known bounded-space bin-packing heuristics, the worst case ratio of REP_3 exactly matches the lower bound of Lee and Lee while using only a constant number of three bins.

We finish this paper by mentioning the following two open problems.

(1) Does repacking help in 2-bounded space on-line bin-packing? The ideas of our algorithm do not work for two active bins. For example, we might receive a sequence of items consisting of five items of size $1/7 + \varepsilon$ and six items of size $1/7$, where ε is some very small appropriately chosen positive real. From this sequence we cannot select a good subset. If the next item to pack is of size greater $2/7$, we *must* close some bin with weight smaller than one.

The 2-bounded Best Fit algorithm of Csirik and Johnson [2] reaches a worst case ratio of 1.7, even without repacking. So the gap is rather narrow.

(2) Can repacking improve the worst case guarantees in higher dimensional bin-packing problems? (see [3] for definitions)

Acknowledgement

Part of this work was carried out while Gerhard Woeginger was visiting József Attila University; Gerhard Woeginger gratefully acknowledges the hospitality of József Attila.

References

[1] Csirik, J., Imreh, B.: On the worst-case performance of the NkF bin-packing heuristic. Acta Cybernetica *9*, 89–105 (1989).
[2] Csirik, J., Johnson, D. S.: Bounded space on-line bin packing: Best ist better than First. Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms, San Francisco, January 1991.
[3] Garey, M.R., Graham, R. L., Johnson, D. S., Yao, A. C. C.: Resource constrained scheduling as generalized bin packing. J. Comb. Th. Ser. A. *21*, 257–298 (1976).
[4] Golomb, S.: On certain non-linear recurring sequences. American Math. Monthly *70*, 403–405 (1963).
[5] Johnson, D. S.: Fast algorithms for bin packing. J. Comput. System Sci. *8*, 272–314 (1974).

[6] Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., Graham, R. L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. *3*, 256–278 (1974).
[7] Lee, C. C., Lee, D. T.: A simple on-line bin-packing algorithm. J. Assoc. Comput. Mach. *32*, 562–572 (1985).
[8] Woeginger, G. J.: Improved space for bounded-space on-line bin-packing. Technical Report No. 187, TU Graz, 1991.

G. Galambos
József Attila University
Department of Applied Computer Sciences
Árpád tér 2, H-6720 Szeged
Hungary

G. J. Woeginger
TU Graz
Institut für Mathematik B
Kopernikusgasse 24, A-8010 Graz
Austria