Improved space for bounded-space, on-line bin-packing

Woeginger, G.

Published in:
SIAM Journal on Discrete Mathematics

DOI:
10.1137/0406045

Published: 01/01/1985

Citation for published version (APA):
IMPROVED SPACE FOR BOUNDED-SPACE, ON-LINE BIN-PACKING*

GERHARD WOEGINGER†

Abstract. The author presents a sequence of linear-time, bounded-space, on-line, bin-packing algorithms that are based on the "HARMONIC" algorithms H_k introduced by Lee and Lee [J. Assoc. Comput. Mach., 32 (1985), pp. 562-572]. The algorithms in this paper guarantee the worst case performance of H_k, whereas they only use O(\log \log k) instead of k active bins. For k ≥ 6, the algorithms in this paper outperform all known heuristics using k active bins. For example, the author gives an algorithm that has worst case ratio less than 17/10 and uses only six active bins.

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

AMS subject classifications. 90B35, 90C27

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 ∈ (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. Since the problem of finding an optimal packing is NP-hard, research has concentrated on approximation algorithms that find near-optimal packings.

Let 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 performance of A, denoted by r(A), is defined as

\[ \lim_{\text{OPT(L)} \to \infty} \sup_{L} \frac{A(L)}{OPT(L)}. \]

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

The 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. Essentially, only the following three types of bounded-space, on-line, bin-packing heuristics have been studied:

(i) The Next-k-Fit (NF_k, k ≥ 2) introduced in [6] simply puts an item a_i into the lowest indexed of k active bins into which it will fit. If no active bin has room for a_i, the lowest-indexed active bin is closed, and a_i is put into a new opened bin. Cseik and Imreh [2] and Mao [8] proved that r(NF_k) = 17/10 + 3/(10k - 10) holds;

(ii) The k-bounded Best Fit (BBF_k, k ≥ 2) introduced in [4] always places an item into the fullest active bin into which it will fit. If no active bin has enough room, a new bin is started, and the fullest active bin is closed. Cseik and Johnson [4] showed in a very sophisticated proof that, independently of the value of k, r(BBF_k) = 17/10 holds;

* Received by the editors June 12, 1991; accepted for publication (in revised form) September 11, 1992.
Part of this work was carried out while the author was visiting József Attila University. This work was partially supported by the Christian Doppler Laboratorium für Diskrete Optimierung.
† TU Graz, Institut für Theoretische Informatik, Klosterwiesgasse 32/II, A-8010 Graz, Austria, (gwoegi@figids02.tu-graz.ac.at).
(iii) The HARMONIC algorithm \( H_k \) [7] is based on a special nonuniform partition of the interval \((0, 1]\) into \( k \) subintervals (where the partitioning points are \( 1/2, 1/3, \ldots, 1/k \)). To each of these subintervals, there corresponds one active bin, and only items belonging to this subinterval are packed into this bin. If some item does not fit into its assigned bin, this bin is closed, and a new bin is used. Lee and Lee [7] analysed the worst case ratio of \( H_k \). They showed that, as \( k \) tends to infinity, \( r(H_k) \) tends to \( h_\infty \approx 1.69103 \).

A summary of the worst case ratios of these heuristics for some small values of \( k \) is given in Table 1. Lee and Lee [7] also showed that for any \( k \)-bounded-space, on-line, bin-packing algorithm \( A \), \( r(A) \geq h_\infty \) must hold. That means that, asymptotically, \( H_k \) is optimal.

In this paper, we present an on-line bin-packing algorithm, called SIMPLIFIED HARMONIC\( k \), or \( SH_k \). By using a better partition of the interval \((0, 1]\) than \( H_k \) does, we get a worst case performance of approximately \( h_\infty + 10^{-5} \) while using only nine active bins! To reach this worst case performance, \( H_k \) had to use 43 active bins, whereas \( NF_k \) and \( BBF_k \) cannot even come beneath \( 17/10 \). Generally, \( SH_k \) has a worst case performance that the HARMONIC algorithm cannot reach by using a number of active bins less than doubly exponential in \( k \).

Furthermore, our heuristic \( SH_k \) has worst case ratio beneath \( 17/10 \) while using only six active bins. This contradicts a conjecture of Csirik [1].

The paper is organized as follows. Sections 2 and 3 present the results on \( SH_k \) for the case where \( k = 3m \). Section 4 extends these results to the other values of \( k \), and § 5 gives the discussion.

2. The simplified harmonic algorithm. The following sequence (introduced by Golomb [5]) is essential in the definition and in the analysis of our algorithm:

\[
t_1 = 2, \\
t_{i+1} = t_i(t_i - 1) + 1 \quad \text{for} \quad i \geq 1.
\]

We will define the algorithm \( SH_k \) only for \( k = 3m, m \geq 1 \). We fix the value of \( m \) for this and the next section and consider the following partition \( \mathcal{P}_k \) of the unit-interval into

<table>
<thead>
<tr>
<th>( k )</th>
<th>( NF_k )</th>
<th>( BBF_k )</th>
<th>( H_k )</th>
<th>( SH_k )</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.00000</td>
<td>1.70000</td>
<td>2.00000</td>
<td>2.00000</td>
<td>1.70000</td>
</tr>
<tr>
<td>3</td>
<td>1.85000</td>
<td>1.70000</td>
<td>1.75000</td>
<td>1.75000</td>
<td>1.70000</td>
</tr>
<tr>
<td>4</td>
<td>1.80000</td>
<td>1.70000</td>
<td>1.72222</td>
<td>1.72222</td>
<td>1.70000</td>
</tr>
<tr>
<td>5</td>
<td>1.77500</td>
<td>1.70000</td>
<td>1.70833</td>
<td>1.70000</td>
<td>1.70000</td>
</tr>
<tr>
<td>6</td>
<td>1.76000</td>
<td>1.70000</td>
<td>1.70000</td>
<td>1.69444</td>
<td>1.69444</td>
</tr>
<tr>
<td>7</td>
<td>1.75000</td>
<td>1.70000</td>
<td>1.69444</td>
<td>1.69388</td>
<td>1.69388</td>
</tr>
<tr>
<td>8</td>
<td>1.74286</td>
<td>1.70000</td>
<td>1.69388</td>
<td>1.69106</td>
<td>1.69106</td>
</tr>
<tr>
<td>9</td>
<td>1.73750</td>
<td>1.70000</td>
<td>1.69345</td>
<td>1.69104</td>
<td>1.69104</td>
</tr>
<tr>
<td>42</td>
<td>1.70732</td>
<td>1.70000</td>
<td>1.69106</td>
<td>1.69103</td>
<td>1.69103</td>
</tr>
<tr>
<td>43</td>
<td>1.70714</td>
<td>1.70000</td>
<td>1.69103</td>
<td>1.69103</td>
<td>1.69103</td>
</tr>
<tr>
<td>( \infty )</td>
<td>1.70000</td>
<td>1.70000</td>
<td>1.69103</td>
<td>1.69103</td>
<td>1.69103</td>
</tr>
</tbody>
</table>
\( k = 3m \) subintervals:

\[
A = (1/2, 1], \\
B_i = (1/t_i, 1/(t_i - 1)] \quad \text{for } i = 2 \ldots m + 1, \\
C_i = (1/(t_i + 1), 1/t_i] \quad \text{for } i = 2 \ldots m, \\
D_i = (1/(t_{i+1} - 1), 1/(t_i + 1)] \quad \text{for } i = 2 \ldots m, \\
E = (0, 1/t_{m+1}].
\]

The algorithm \( \text{SH}_k \) simply proceeds as follows. For each of the \( k \) subintervals, it keeps a separate active bin. In this bin, only items belonging to the corresponding subinterval are packed. Now, if the algorithm receives a new item \( a_i \) to pack, it first classifies \( a_i \) according to the partition \( \mathcal{P}_k \). Then it tries to pack \( a_i \) into its corresponding active bin. If there is not enough room in the bin, this bin is closed, a new active bin is opened, and \( a_i \) is put into it.

Since item classification can be done in \( O(\log k) \) time and there are only \( k \) active bins at any time, the algorithm runs in \( O(n \log k) \). Hence, if we take \( k \) to be constant, the time complexity is linear.

3. Worst case analysis of simplified harmonic. We define below a weighting function \( W_k(x) \), which we prove has the following two properties. If \( W_k(L) \) is the cumulative weight of the pieces in \( L \), then (i) the length of the \( \text{SH}_k \)-packing of \( L \) cannot exceed \( W_k(L) + k \) and (ii) \( W_k(L) \) cannot exceed \( \Gamma_k \) times the length of an optimum packing, where (remember that \( k = 3m \))

\[
\Gamma_k = \sum_{i=1}^{m} \frac{1}{t_i - 1} + \frac{t_{m+1}}{(t_{m+1} - 1)^2}
\]

holds. From this, it immediately follows that \( \text{SH}_k(L) - k \leq W_k(L) \leq \Gamma_k \text{OPT}(L) \), and this yields \( r(\text{SH}_k) \leq \Gamma_k \). Define the weighting function \( W_k(x) \) as follows:

\[
W_k(x) = x + 1/2 \quad \text{for } 1/2 < x \\
= x + \frac{1}{t_{i+1} - 1} \quad \text{for } 1/t_i < x \leq 1/(t_i - 1) \text{ and } 2 \leq i \leq m + 1 \\
= t_i + \frac{1}{t_i} \cdot x \quad \text{for } 1/(t_{i+1} - 1) < x \leq 1/t_i \text{ and } 2 \leq i \leq m \\
= \frac{t_{m+1}}{t_{m+1} - 1} \cdot x \quad \text{for } x \leq 1/t_{m+1}.
\]

An illustration for this weighting function and for the partition of the unit-interval in the case where \( k = 6 \) is given in Table 2. The following observation immediately follows from the definition of the weighting function.

Observation 1. For \( i \leq m \) and \( x \leq 1/t_i \), \( W_k(x)/x \leq (t_i + 1)/t_i \) holds.

Claim 1. \( W_k(L) \geq \text{SH}_k(L) - k \).

Proof. We show that, in the \( \text{SH}_k \)-packing, every closed bin \( B \) has weight at least 1. Together with the \( k \) last active bins, this implies the claim. We distinguish the following five cases:

(i) The bin \( B \) corresponds to the \( A \)-interval. Then it contains a single item of weight greater than 1;
(ii) The bin $B$ corresponds to some $B_i$-interval, $2 \leq i \leq m + 1$. Then it contains exactly $t_i - 1$ items, each of size greater than $1/t_i$. Consequently, the total weight of $B$ is at least

$$(t_i - 1) \cdot W_k \left( \frac{1}{t_i} + e \right) > (t_i - 1) \cdot \left( \frac{1}{t_i} + \frac{1}{t_i+1-1} \right) = 1;$$

(iii) The bin $B$ corresponds to some $C_i$-interval, $2 \leq i \leq m$. Then it contains exactly $t_i$ items, each of size greater than $1/(t_i + 1)$. This gives a total weight of at least

$$t_i \cdot W_k \left( \frac{1}{t_i+1} \right) = t_i \cdot \frac{t_i+1}{t_i} \cdot \frac{1}{t_i+1} = 1;$$

(iv) The bin $B$ corresponds to some $D_i$-interval, $2 \leq i \leq m$. Since the bin was closed, some item in $D_i$ did not fit into it. Therefore $B$ is at least $t_i/(t_i + 1)$ full and as the weight function is linear on this interval, the total weight is at least

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

(v) The bin $B$ corresponds to the $E$-interval. Analogously to (iv), we see that $B$ is at least $(t_{m+1} - 1)/t_{m+1}$ full and that the total weight is at least $1$.

**CLAIM 2.** In any packing of $L$, the weight of any bin is at most $\Gamma_k$. Hence, $W_k(L) \leq \Gamma_k \text{OPT}(L)$ holds.

**Proof.** Consider some fixed bin $B$ that contains items $q_1 \geq q_2 \geq \cdots \geq q_n$. We distinguish two cases.

(i) $q_i \in (1/t_i, 1/(t_i - 1)]$ for $i = 1 \ldots m$. We denote by $Q$ the sum $\sum_{i=m+1}^{n} q_i$. Obviously, $Q \leq 1/(t_{m+1} - 1)$ holds. Now

$$W_k(B) = \sum_{i=1}^{n} W_k(q_i) = \sum_{i=1}^{m} \left( q_i + \frac{1}{t_i+1-1} \right) + \sum_{i=m+1}^{n} W_k(q_i) \leq 1 - Q + \sum_{i=1}^{m} \frac{1}{t_i+1-1} + \frac{t_{m+1}}{t_{m+1} - 1} \cdot Q.$$

It is easy to see that the latter expression becomes maximum when $Q$ takes its maximum value $1/(t_{m+1} - 1)$; in this case, the expression is exactly $\Gamma_k$.

(ii) Suppose that $r \leq m$ is the least $i$ such that $q_i \in (1/t_i, 1/(t_i - 1)]$, and hence $q_r \leq 1/t_r$. We denote by $Q$ the sum $\sum_{i=r}^{n} q_i$. Obviously, $Q < 1/(t_r - 1)$ holds, and, by Observation 1, the total weight of all elements $q_r \ldots q_n$ is less than or equal to
(t_r + 1)Q / t_r. Similarly as in (i), this yields

\[ W_k(B) \leq 1 - Q + \sum_{i=1}^{r-1} \frac{1}{l_{i+1} - 1} + \frac{t_r + 1}{t_r}Q \]
\[ \leq \sum_{i=1}^{r-1} \frac{1}{l_i - 1} + \frac{t_r + 1}{l_r(t_r - 1)} = \sum_{i=1}^{r+1} \frac{1}{l_i - 1} < \Gamma_k, \]

and the proof of Claim 2 is complete.

**Lemma 1.** For \( k = 3m \), the asymptotic worst case ratio of the heuristic SIMPLIFIED HARMONIC \( \Gamma_k \).

**Proof.** Claims 1 and 2 imply that \( r(SH_k) \leq \Gamma_k \) holds. To show that the bound is tight, we present a family of lists \( L_n \). We define

\[ \alpha_m = (t_m + 1)(t_m + 1 - 2). \]

Now let \( n \) be a multiple of \( \alpha_m \). The optimum packing of our list \( L_n \) will use \( n + 1 \) bins, and the \( SH_k \)-packing will use \( n \Gamma_k \) bins. We choose two very small positive reals \( \varepsilon \) and \( \delta \) such that

\[ (m + 1)\alpha_m \cdot \varepsilon + \delta \leq \alpha_m / n. \]

We define \( L_n \) by giving its optimum packing. In this packing, we have bins of the following contents:

- \( n / \alpha_m \) times a bin that contains
  \[ 1 / t_i + \varepsilon \text{ (for } i = 1 \ldots m), \ 1 / t_m + 1, \ 1 / (t_{m+2} - 1) - (m + 1)\varepsilon; \]

- \( (\alpha_m - 1)n / \alpha_m \) times a bin that contains
  \[ 1 / t_i + \varepsilon \text{ (for } i = 1 \ldots m), \ 1 / t_m + 1 + \varepsilon, \ 1 / (t_{m+2} - 1) - (m + 1)\varepsilon; \]

- a single bin containing
  \[ (n / \alpha_m)\text{-times an item of size } (m + 1)\alpha_m \cdot \varepsilon + \delta. \]

In the \( SH_k \)-packing, the packing of the items of size \( 1 / t_i + \varepsilon \), \( 1 \leq i \leq m \) is easy to analyse. Independently of their ordering, they use exactly \( n / (t_i - 1) \) bins. Analogously, we see that the items of size \( 1 / t_m + 1 + \varepsilon \) are packed into \( (\alpha_m - 1) / \alpha_m \cdot n / (t_{m+1} - 1) \) bins. Thus, the only interesting items are the items that are \( \leq 1 / (t_{m+1}) \text{ (i.e., the } E\text{-items).} \)

These are given to \( SH_k \) in \( n \alpha_m \) "packages" of the following type:

\[ \frac{1}{t_{m+1}}, \frac{1}{t_{m+2} - 1} - (m + 1)\varepsilon, \ (m + 1)\alpha_m \cdot \varepsilon + \delta. \]

We show that \( SH_k \) puts each package into a separate bin. This holds, since the total size of a package is exactly \( (t_{m+1} - 1) / t_{m+1} + \delta \). Hence, when the first item of the next package arrives, it does not fit into the active bin. Consequently, the bin is closed, and the next package is treated in the same way. Summarizing, \( SH_k \) uses a total number of

\[ \sum_{i=1}^{m} \frac{n}{l_i - 1} + \frac{\alpha_m - 1}{\alpha_m} \cdot \frac{n}{l_{m+1} - 1} + \frac{n}{\alpha_m} \]

bins, and this number is equal to \( n \Gamma_k \). □
4. Main results. In this section, we extend the results of the preceding sections to the cases where $k = 3m + 1$ and $k = 3m - 1$. The underlying set $T$ of partitioning points of the unit-interval is given by

$$T = \bigcup_{i=1}^{\infty} \left\{ \frac{1}{t_i + 1}, \frac{1}{t_i}, \frac{1}{t_i - 1} \right\}.$$ 

To define the algorithm $SH_k$ for general $k$, we take the $k - 1$ largest values in $T$. These values partition the interval $(0, 1)$ into $k$ subintervals. $SH_k$ keeps for each subinterval a separate active bin and proceeds exactly as in the preceding sections. It is easy to verify that for $k = 3m$ this indeed leads to our old algorithm. Before we can state our main theorem, we give the following definitions, for $m \geq 1$:

$$\Gamma_{3m-1} = \sum_{i=1}^{m} \frac{1}{t_i - 1} + \frac{1}{t_{m+1} - 2},$$

$$\Gamma_{3m} = \sum_{i=1}^{m} \frac{1}{t_i - 1} + \frac{t_{m+1}}{(t_{m+1} - 1)^2},$$

$$\Gamma_{3m+1} = \sum_{i=1}^{m} \frac{1}{t_i - 1} + \frac{t_{m+1}^2 + t_{m+1} + 1}{t_{m+1}(t_{m+1} - 1)}.$$

**THEOREM 1.** For $k \geq 2$, the asymptotic worst case ratio of the heuristic SIMPLIFIED HARMONIC$_k$ is $\Gamma_k$.

The proofs are analogous to the proofs of Claims 1 and 2 and to the proof of Lemma 1. Essentially, we use the same weighting function again. The only modification concerns elements $x$ in the smallest interval $(0, a]$; these elements always get weight $x/(1 - a)$. The details are left to the reader as an exercise. Some values of $\Gamma_k$ for some small $k$ are given in the fourth column of Table 1.

Finally, we compare the behaviour of the heuristics $H_k$ and $SH_k$. Theorem 2 in [7] states that, for $k = t_{m+1} - 1$,

$$r(H_k) = \sum_{i=1}^{m} \frac{1}{t_i - 1} + \frac{1}{t_{m+1} - 2}.$$ 

This value is equal to our $\Gamma_{3m-1}$. Consequently, HARMONIC using $t_{m+1} - 1$ active bins and SIMPLIFIED HARMONIC using $3m - 1$ active bins achieve the same asymptotic worst case ratio. As the $t_i$ grow doubly exponentially, the following theorem holds.

**THEOREM 2.** To achieve the worst case performance of heuristic HARMONIC$_k$ with $k$ active bins, the heuristic SIMPLIFIED HARMONIC only has to use $O(\log \log k)$ active bins.

5. Discussion. In this paper, we derived a sequence of new $k$-bounded-space, online, bin-packing algorithms called SIMPLIFIED HARMONIC$_k$. For $k \geq 6$, the worst case behaviour of our algorithms outperforms all known heuristics using $k$ active bins. For $k \leq 4$, the best-known algorithms are the BBF$_k$ due to Csirik and Johnson. For $k = 5$, BBF$_5$ and $SH_5$ both have the same worst case performance.

The average performance of SIMPLIFIED HARMONIC$_k$ suffers from the usual drawback of harmonic algorithms: For $L_n$, a random list of $n$ items with sizes chosen independently from a uniform distribution, the average value $H_k(L)/OPT(L)$ approaches 1.28987 as $k$ tends to $\infty$ (see [3]), whereas the average value of $NF_k(L)/OPT(L)$ empirically approaches 1. Computational experiments performed on large item lists indicate that, in the average case, $SH_k$ performs as poorly as $H_k$ does.
There remains a number of (seemingly hard) open questions.

(1) What is the best possible worst case performance of any on-line, bin-packing heuristic using 2-bounded space? (BBF$_2$ achieves a worst case ratio of $17/10$.)

(2) What is the smallest $k$ such that there exists an on-line, bin-packing heuristic using $k$-bounded space with asymptotic worst case ratio strictly less than $17/10$? (SH$_6$ comes beneath $17/10$ by using 6-bounded space.)

(3) If we only consider algorithms that pack the items by Next-Fit according to some fixed partition of $(0, 1]$ into $k$ subintervals, which partition gives the best worst case ratio? (It is easy to see that, for $k = 1$ and $k = 2$, in this case the best possible worst case performance is 2, but for $k \geq 3$ no tight bounds are known.)

**Acknowledgments.** I gratefully acknowledge the hospitality of József Attila. Furthermore, I thank János Csirik, Gábor Galambos, and Hannes Hassler for several helpful discussions.

**REFERENCES**