Ordered Strip Packing

Ordered Strip Packing*

K. Buchin[1], D. Kosolobov[2], W. Sonke[1], B. Speckmann[1], and K. Verbeek[1]

[1] TU Eindhoven, The Netherlands
{k.a.buchin,w.m.sonke,b.speckmann,k.a.b.verbeek}@tue.nl
[2] Ural Federal University, Russia
dkosolobov@mail.ru

Abstract. We study an ordered variant of the well-known strip packing problem, which is motivated by applications in visualization and typography. Our input consists of a maximum width W and an ordered list of n blocks (rectangles). The goal is to pack the blocks into rows (not exceeding W) while obeying the given order and minimizing either the number of rows or the total height of the drawing. We consider two variants: (1) non-overlapping row drawing (NORD), where distinct rows cannot share y-coordinates, and (2) overlapping row drawing (ORD), where consecutive rows may overlap vertically. We present an algorithm that computes the minimum-height NORD in $O(n)$ time. Further, we study the worst-case tradeoffs between the two optimization criteria—number of rows and total height—for both NORD and ORD. Surprisingly, we show that the minimum-height ORD may require $\Omega(\log n / \log \log n)$ times as many rows as the minimum-row ORD. The proof of the matching upper bound employs a novel application of information entropy.

Keywords: Linear layouts · Packing

1 Introduction

Packing problems arise in many practical applications and have hence been studied extensively in the literature. Two well-known classes of 2D packing problems are *bin packing* and *strip packing*. In both problems, the aim is to pack (without rotation) a set of *blocks* (rectangles) into a shape such that they are internally disjoint. The goal of bin packing is to pack the blocks into rectangular *bins* of the same (given) size while minimizing the number k of bins used. The goal of strip packing is to pack the blocks into a *strip* of fixed width W and infinite height while minimizing the height H of the strip used.

In this paper we introduce a new *ordered* variant of strip packing where we are also given an order on the blocks which the packing must obey. Such orders arise naturally in applications such as typography or visualization: When dividing the text in a paragraph into lines, it is obviously necessary to retain the word order (see, for example, the word wrapping algorithm by Knuth and Plass still

* Willem Sonke, Bettina Speckmann and Kevin Verbeek are partially supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208 (W.S. and B.S.) and no. 639.021.541 (K.V.).

used in TEX [4]). Or consider the visual layout of linear sequences such as time lines or industrial processes: to make better use of the available screen space the linear layouts need to be "folded" into a strip (the screen) while keeping the sequence intact (see Fig. 1).

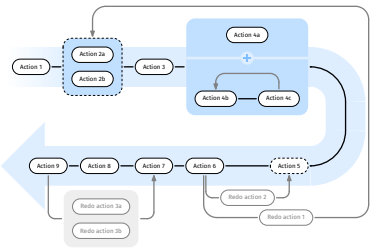


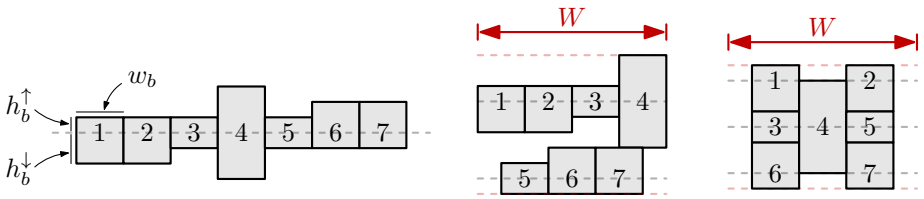
Fig. 1. Process visualization [9].

Problem statement. Our input consists of an ordered list B of n blocks and a maximum width W. Each block $b = (w_b, h_b^{\uparrow}, h_b^{\downarrow})$ is specified by its width w_b, its top height $h_b^{\uparrow}$, and its bottom height $h_b^{\downarrow}$. The goal is to place the blocks on rows. Each row consists of a horizontal line called the *spine*, on which blocks are aligned vertically (see Fig. 2a). The top and bottom heights $h_b^{\uparrow}$ and $h_b^{\downarrow}$ specify how much block b sticks out above and below the spine, respectively. A solution consists of a set of y-coordinates for the spines, and, for every block, a row and an x-coordinate. More precisely, the placement (*drawing*) of the blocks must follow these rules:

1. All blocks are interior disjoint.
2. All blocks lie completely in the x-range $[0, W]$.
3. The height of a block b above (below) the assigned spine is $h_b^{\uparrow}$ ($h_b^{\downarrow}$).
4. The interior of a block may not overlap with the spine of another row.
5. The order of the blocks from top to bottom, and then from left to right, must coincide with the specified order in the input (see Fig. 2b).

We consider two natural optimization questions: (1) minimizing the total height of the drawing, and (2) minimizing the number of rows. Furthermore, we distinguish two versions: In a *non-overlapping rows drawing (NORD)*, blocks on different rows may not share the same y-coordinates (see Fig. 2b). In an *overlapping rows drawing (ORD)* this restriction is omitted (see Fig. 2c). Minimizing the total height of an ORD is NP-hard, while a NORD of minimum total height can be computed in $O(n^2)$ time [9]. Furthermore, NORDs and ORDs that minimize the number of rows can easily be computed greedily in polynomial time.

Related work. 2D bin packing and strip packing have been widely studied. Both problems are NP-hard, since 1D bin packing is already NP-hard [5]. Hence,



(a) List of blocks. (b) NORD. (c) ORD.

Fig. 2. Minimum-height NORD and ORD of a list of blocks.

research has focused on approximation algorithms. See, for example, the surveys by Coffman *et al.* [1] and Lodi *et al.* [7]. Some online approximation algorithms for strip packing are *level-based*, that is, they layout blocks on horizontal *levels*, akin to rows in our problem. For example, the *next-fit* algorithm [3] greedily puts blocks on a level in the input order, until the next block does not fit anymore. This algorithm therefore produces minimum-row NORDs.

The ordered strip packing problem for NORDs is related to the *word wrap problem*, where the aim is to split a list of words into lines such that some quality measure, usually defined in terms of line lengths, is maximized. Knuth and Plass proposed an algorithm for word wrapping, which is still in use in TEX [4].

Ordered strip packing for NORDs can be considered a special case of the *least-weight subsequence problem*, which was introduced by Hirschberg and Larmore [2] as a generalization of the word wrap problem. Given a weight function $f : \{1, \ldots, n\} \times \{1, \ldots, n\} \to \mathbb{R}$, they ask for a minimum-weight subsequence S of $[1, 2, \ldots, n]$, starting with 1 and ending with n. Here the weight of a sequence $\{s_i\}_{i=0}^{n}$ is given by $\sum_{i=1}^{n} f(s_{i-1}, s_i)$. Hirschberg and Larmore present a straightforward $O(n^2)$-time algorithm, which can be improved to $O(n \log n)$ time if f is concave (that is, $f(i,k) + f(j,l) \leq f(i,l) + f(j,k)$ for all $i \leq j < k \leq l$), and to $O(n)$ time if the weight function satisfies an additional condition. Wilber [10] improved this to an $O(n)$-time algorithm without requiring the additional condition. However, the weight function for minimum-height NORDs is not concave, and this method thus cannot be used.

Results and organization. In Section 2 we present a new algorithm that can compute a minimum-height NORD with n blocks in $O(n)$ time, improving upon the best known existing algorithm [9] that runs in $O(n^2)$ time. In Section 3 we study the tradeoffs between the two optimization criteria: how is the height of the drawing affected if we minimize the number of rows, and how is the number of rows affected if we minimize the height. We present upper and lower bounds for these tradeoffs. The most interesting and most surprising bound shows that a minimum-height ORD may require $\Omega(\log n / \log \log n)$ times as many rows as the minimum-row ORD. We prove that this bound is tight using a novel application of the information entropy function. We believe that this new way of using entropy is of independent interest, and may have further applications in packing problems. Omitted proofs can be found in the full version of the paper.

2 Computing NORDs

We first present a basic quadratic-time algorithm for computing minimum-height NORDs (Section 2.1). We improve the running time to linear, first for vertically-centered blocks (Section 2.2), and then for the general case (Section 2.3).

2.1 Basic algorithm

We need to determine for each block whether it starts a new row or not. To do this efficiently, we use the following observation. In a k-row drawing, we call the last block on row $k-1$ the *separating block*.

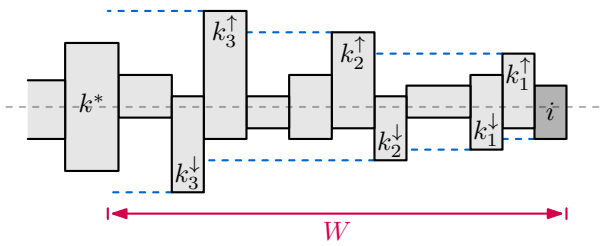


Fig. 3. Illustration of Lemma 1, which allows us to consider only a subset of the blocks to end row $k-1$. Blocks in $K_i^\uparrow$ and $K_i^\downarrow$ are marked here as $k_j^\uparrow$ and $k_j^\downarrow$, respectively.

Lemma 1. *Let B be a list of blocks. A minimum-height NORD for B exists such that the separating block either (1) has larger top height than all blocks on the last row, or (2) has larger bottom height than all blocks on the last row, or (3) together with the blocks on the last row, has total width larger than W.*

We process the blocks in order, incrementally creating minimum-height NORDs for the first $1, \ldots, n$ blocks. Assume that we are constructing a minimum-height NORD for the first i blocks. By Lemma 1 we only consider drawings in which the separating block satisfies conditions (1), (2), or (3). Let k^* be the smallest integer such that blocks $k^*+1, \ldots, i$ have total width at most W. Blocks $1, \ldots, k^*-1$ cannot serve as the separating block, because that would overfill the last row. The blocks k after k^* that satisfy condition (1) are those with successively larger top heights (starting at block i going backwards to k^*); they form a "staircase" pattern. We call the set of these blocks the *top candidate set* $K_i^\uparrow$ for blocks $1, \ldots, i$. Similarly, the blocks k after k^* that satisfy condition (2) form a staircase of increasing bottom heights; we call this the *bottom candidate set* $K_i^\downarrow$ (see Fig. 3). After computing $K_i^\uparrow$, $K_i^\downarrow$, and k^*, we compute

$$T[i] = \min_{k \in K_i^\uparrow \cup K_i^\downarrow \cup \{k^*\}} \big(T[k] + h_{\text{row}}(k+1, i)\big),$$

where $h_{\text{row}}(k+1,i) := h^\uparrow_{\text{row}}(k+1,i) + h^\downarrow_{\text{row}}(k+1,i)$, $h^\uparrow_{\text{row}}(k+1,i) = \max\{h_j^\uparrow \mid j = k+1, \ldots, i\}$, and $h^\downarrow_{\text{row}}(k+1,i) = \max\{h_j^\downarrow \mid j = k+1, \ldots, i\}$ are the height, top height, and bottom height of a row with blocks $k+1, \ldots, i$. It follows from Lemma 1 that $T[i]$, for $i = 1, \ldots, n$, is the height of a minimum-height NORD for the blocks $1, \ldots, i$, provided $T[0] = 0$. As sets $K_i^\uparrow$ and $K_i^\downarrow$ may have linear size, computing $T[i]$ takes $O(i)$ time, resulting in a total running time of $O(n^2)$.

2.2 Algorithm for vertically centered blocks

Next we consider the special case when all blocks are vertically centered, i.e., $h_i^\uparrow = h_i^\downarrow$ for all blocks i. We denote the total height of block i by h_i $(= h_i^\uparrow + h_i^\downarrow)$. Since the top and bottom candidate sets $K_i^\uparrow$ and $K_i^\downarrow$ coincide in this case, we denote them simply as K_i.

Fix i and the corresponding k^* defined as before. Let $k_1, \ldots, k_s$ denote all blocks from K_i in the right-to-left order, so that $k^* < k_s < \cdots < k_1 < i$. Recall that K_i consists exactly of all blocks k between k^* and i such that $h_k > \max\{h_{k+1}, h_{k+2}, \ldots, h_i\}$ and, thus, $h_{k_s} > \ldots > h_{k_1}$. Therefore, the value $h_{\text{row}}(k+1, i)$, for $k \in K_i \cup \{k^*\}$, can be determined as follows:

$$h_{\text{row}}(k_1 + 1, i) = h_i;$$
$$h_{\text{row}}(k_j + 1, i) = h_{k_{j-1}}, \text{ for } 1 < j \leq s;$$
$$h_{\text{row}}(k^* + 1, i) = h_{k_s}.$$

Instead of a naïve linear computation of the minimum as in the quadratic algorithm, we store K_i in a so-called *mindeque* [6]: a deque that supports the standard insertions and deletions in constant amortized time and that can compute the minimum of the values assigned to its elements in constant time. We assign to each $k_j \in K_i$ the value $T[k_j] + h_{k_{j-1}}$, for $j > 1$, and $T[k_1] + h_i$, for $j = 1$. Therefore, one can calculate $T[i] = \min_{k \in K_i \cup \{k^*\}}(T[k] + h_{\text{row}}(k+1, i))$ as the minimum of $T[k^*] + h_{k_s}$ and all values assigned to the deque blocks. Thus, $T[i]$ can be calculated in $O(1)$ time.

It remains to show that the mindeque storing K_i can be maintained with $O(n)$ insertions and deletions. For this, we describe how to modify k^* and transform K_i into K_{i+1}. First, k^* is updated by consecutive increments until $h_{k^*+1} + \cdots + h_{i+1} \leq W$. From K_i we have to remove the blocks that are to the left of the new k^*. Thus, we dequeue the leftmost blocks $k_s, k_{s+1}, \ldots, k_t$ with $k_s < \cdots < k_t \leq k^* < k_{t+1}$ for the new k^* (or $k_t = k_1$ if $k_1 \leq k^*$). Denote by K'_i the updated set K_i. As k^* does not decrease, in total at most $O(n)$ such deletions are performed.

A block k belongs to K_{i+1} iff $h_k > \max_{k<j\leq i+1} h_j$ and $k^* < k < i+1$. However, K'_i contains exactly all k such that $h_k > \max_{k<j\leq i} h_j$ and $k^* < k < i$. To obtain K_{i+1} from K'_i we still need to remove all $k \in K'_i$ with $h_k \leq h_{i+1}$ and then insert the block i if $h_i > h_{i+1}$. Now since $h_{k_s} > \cdots > h_{k_1}$, the blocks to be removed are rightmost in K'_i. Thus, we can dequeue blocks $k_p, \ldots, k_1$ until $h_{k_{p+1}} > h_{i+1} \geq h_{k_p}$. Thus, all modifications can be performed by deque operations.

As described above, any block is added at most once to the mindeque, and therefore any block is removed at most once. Thus, we perform $O(n)$ mindeque operations in total and the overall running time is linear. Since the mindeque data structure can identify an element on which the minimum is attained, a minimum-height NORD can be reconstructed via standard backtracking.

Theorem 1. *A minimum-height NORD of n vertically-centered blocks can be computed in $O(n)$ time using a mindeque.*

2.3 General linear algorithm

Consider the general case where $h_i^{\uparrow}$ and $h_i^{\downarrow}$ can differ. Fix i and k^* and denote by $k_1^{\uparrow}, \ldots, k_s^{\uparrow}$ and $k_1^{\downarrow}, \ldots, k_t^{\downarrow}$ all blocks of $K_i^{\uparrow}$ and $K_i^{\downarrow}$ in the right-to-left order. By definition, $K_i^{\uparrow}$ contains exactly all blocks k such that $k^* < k < i$ and $h_k^{\uparrow} >$

$\max_{k<j\le i} h^{\uparrow}_j$; similarly, $k \in K^{\downarrow}_i$ iff $k \in (k^*, i)$ and $h^{\downarrow}_k > \max_{k<j\le i} h^{\downarrow}_j$. Therefore, as in Section 2.2, $K^{\uparrow}_i$ can be maintained in a deque and transformed into $K^{\uparrow}_{i+1}$ by removing the leftmost blocks 'swept' by k^* and the rightmost blocks k with $h^{\uparrow}_k \le h^{\uparrow}_{i+1}$, and by inserting i if $h^{\uparrow}_i > h^{\uparrow}_{i+1}$. In total, this requires $O(n)$ operations when i passes from 1 to n; $K^{\downarrow}_i$ can be processed analogously. It is unclear, however, how to efficiently use these two deques, since maintaining the row heights associated with the blocks in $K^{\uparrow}_i$ and $K^{\downarrow}_i$ requires a more subtle approach.

We split $K^{\uparrow}_i$ into maximal contiguous subsequences that do not interleave $K^{\downarrow}_i$: $(k^{\uparrow}_{j_\ell+1}, k^{\uparrow}_{j_\ell+2}, \dots, k^{\uparrow}_{j_{\ell+1}})^{p}_{\ell=0}$, where for all $k \in K^{\downarrow}_i$ either $k > k^{\uparrow}_{j_\ell+1}$ or $k^{\uparrow}_{j_{\ell+1}} \ge k$ and $0 = j_0 < \dots < j_p = s$. Likewise, $K^{\downarrow}_i$ is split into $(k^{\downarrow}_{j'_\ell+1}, k^{\downarrow}_{j'_\ell+2}, \dots, k^{\uparrow}_{j'_{\ell+1}})^{q}_{\ell=0}$ non-interleaving with $K^{\uparrow}_i$, where $0 = j'_0 < \dots < j'_q = t$. The subsequences are arranged from right to left and each of them is stored in a separate mindeque d_ℓ (the values assigned to the blocks for d_ℓ are defined below) in this order:

$$\underbrace{k^{\uparrow}_1, k^{\uparrow}_2, \dots, k^{\uparrow}_{j_1}}_{d_1}, \underbrace{k^{\downarrow}_1, k^{\downarrow}_2, \dots, k^{\downarrow}_{j'_1}}_{d_2}, \underbrace{k^{\uparrow}_{j_1+1}, k^{\uparrow}_{j_1+2}, \dots, k^{\uparrow}_{j_2}}_{d_3}, \underbrace{k^{\downarrow}_{j'_1+1}, k^{\downarrow}_{j'_1+2}, \dots, k^{\downarrow}_{j'_2}}_{d_4}, \dots$$

The sequence is non-increasing and only adjacent blocks $k^{\uparrow}_{j_\ell}$, $k^{\downarrow}_{j'_{\ell-1}+1}$ or $k^{\downarrow}_{j'_\ell}$, $k^{\uparrow}_{j_\ell+1}$ can coincide in it (such blocks belong to both top and bottom candidate sets). When i increases, some rightmost and leftmost blocks from $K^{\uparrow}_i$ and $K^{\downarrow}_i$ are removed and, in the process, some d_ℓ might be deleted entirely. To retain maximality of the subsequences, we then have to join some d_ℓ. In order to do this efficiently, we store d_ℓ as *catenable mindeques* [6] that can be concatenated in $O(1)$ amortized time. Thus, all the d_ℓ are maintained in $O(n)$ overall time.

For any block k, we have $\max_{k<j\le i} h^{\downarrow}_j = h^{\downarrow}_b$ for $b = \min\{b \in K^{\downarrow}_i \cup \{i\} \colon k < b\}$. Hence, given a deque d_ℓ whose blocks are from $K^{\uparrow}_i$, all its blocks k yield the same value $\max_{k<j\le i} h^{\downarrow}_j$, which we denote a_ℓ: if $\ell = 1$, $a_\ell = h^{\downarrow}_i$; otherwise, $a_\ell = h^{\downarrow}_m$, where m is the leftmost block in $d_{\ell-1}$ that is not in d_ℓ (observe that d_ℓ and $d_{\ell-1}$ can share a block only if d_ℓ contains only one block). Analogously, for $d_\ell \subseteq K^{\downarrow}_i$ the maximum $\max_{k<j\le i} h^{\uparrow}_j$ is the same for all $k \in d_\ell$ and we denote it a_ℓ.

Each mindeque d_ℓ can compute the minimum, denoted $\min d_\ell$, of the values assigned to its blocks. We assign to each $k^{\uparrow}_r \in K^{\uparrow}_i$ (resp., $k^{\downarrow}_r \in K^{\downarrow}_i$) the value $T[k^{\uparrow}_r] + h^{\uparrow}_{k^{\uparrow}_{r-1}}$ (resp., $T[k^{\downarrow}_r] + h^{\downarrow}_{k^{\downarrow}_{r-1}}$), for $r > 1$, and $T[k^{\uparrow}_1] + h^{\uparrow}_i$ (resp., $T[k^{\downarrow}_1] + h^{\downarrow}_i$), for $r = 1$. We store pointers to $d_1, d_2, \dots$ in a mindeque D and assign $a_\ell + \min d_\ell$ to the pointer to d_ℓ; D is easy to maintain along with the deques d_ℓ in $O(n)$ overall time. Recall that $T[i] = \min\{T[k] + h_{\mathrm{row}}(k+1, i) \colon k \in K^{\uparrow}_i \cup K^{\downarrow}_i \cup \{k^*\}\}$. For each $k^{\uparrow}_r$ with $r > 1$, we have $T[k^{\uparrow}_r] + h_{\mathrm{row}}(k^{\uparrow}_r + 1, i) = T[k^{\uparrow}_r] + h^{\uparrow}_{r-1} + \max_{k^{\uparrow}_r < j \le i} h_j$. Observe that for the deque d_ℓ containing $k^{\uparrow}_r$, $\max_{k^{\uparrow}_r < j \le i} h_j = a_\ell$ and the value $T[k^{\uparrow}_r] + h^{\uparrow}_{r-1}$ is assigned to $k^{\uparrow}_r$; we analogously analyze $T[k] + h_{\mathrm{row}}(k+1, i)$ for k equal to $k^{\uparrow}_1$, $k^{\downarrow}_1$, and $k^{\downarrow}_r$ with $r > 1$. Therefore, one can compute $T[i]$ as the minimum of $\min D$ and $T[k^*] + h_{\mathrm{row}}(k^* + 1, i)$, where $h_{\mathrm{row}}(k^* + 1, i) = h^{\uparrow}_s + h^{\downarrow}_t$.

Theorem 2. *A minimum-height NORD of n blocks can be found in $O(n)$ time.*

Table 1. All tradeoffs between minimizing rows and height for NORDs and ORDs.

Type	$\boldsymbol{\alpha(n)}$		$\boldsymbol{\beta(n)}$	
	Lower	Upper	Lower	Upper
NORD	≥ 2	≤ 2	$\geq 3/2$	≤ 2
ORD	≥ 4	≤ 4	$\Omega(\log n / \log\log n)$	$O(\log n / \log\log n)$

3 Optimization tradeoffs

Depending on the application, we may want to either minimize the number of rows of the drawing, minimize the total height, or a combination. To study the effect of the two optimization criteria, we analyze the worst-case tradeoffs between them. More precisely, we consider the worst-case ratio $\alpha(n)$ between the heights of minimum-row and minimum-height drawings (ORD and NORD):

$$\alpha(n) := \sup_{\text{list } B \text{ of } n \text{ blocks}} \frac{\text{height of minimum-row drawing of } B}{\text{height of minimum-height drawing of } B}.$$

Secondly we consider the worst-case ratio $\beta(n)$ between the number of rows of minimum-height and minimum-row drawings:

$$\beta(n) := \sup_{\text{list } B \text{ of } n \text{ blocks}} \frac{\#\text{ rows of minimum-height drawing of } B}{\#\text{ rows of minimum-row drawing of } B}.$$

We assume that the two criteria are optimized lexicographically, for example the minimum-height drawing has the fewest rows among all drawings with the minimum height. All bounds are summarized in Table 1. Proofs for the simple cases can be found in the full version of the paper. Next we analyze the most interesting case, namely the number of rows of a minimum-height ORD.

3.1 Lower bound for minimum-height ORDs

In this section we prove a lower bound on the number of rows of a minimum-height ORD compared to the minimum-row ORD. For any integer $k > 2$, we construct a list of $n = \Theta(k^k)$ blocks such that a minimum-row ORD uses 2 rows, and a minimum-height ORD requires at least $k + 1$ rows (see Fig. 4). Let the width of the drawing be $W = k^k$. Row k contains one block of width k^{k-1}. Row $k-1$ contains k blocks, each of width k^{k-2}. Generally, row i ($1 \leq i \leq k$) contains k^{k-i} blocks of width k^{i-1}. Hence, for each row, the sum of its block widths is exactly k^{k-1}, and the sum of all block widths on rows $1, \ldots, k$ is k^k.

The top height of the blocks on row i is 2^{k-i+1}. The bottom height of the blocks on row i is 2^{k-i}, that is, equal to the top height of the blocks on the next row. However, every k-th block (starting with the first) on each row is *truncated*: it has bottom height 0. This ensures that the top and bottom heights of adjacent rows can fully overlap, so the vertical distance between the spines

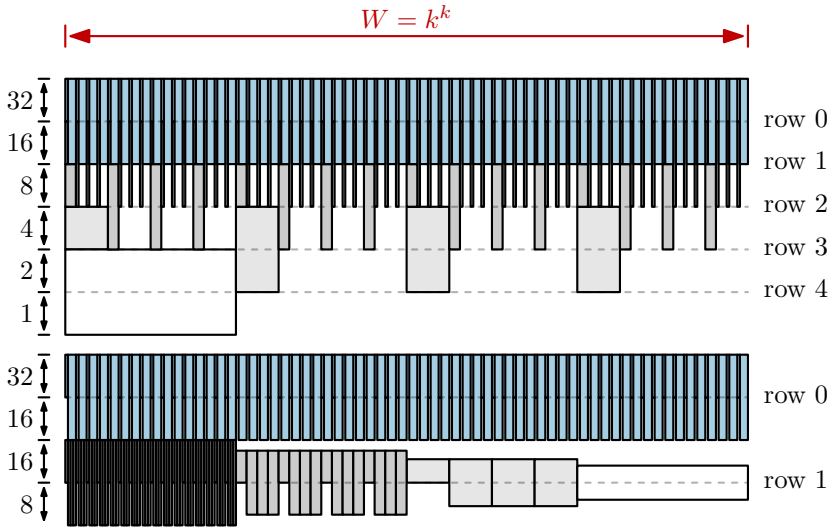


Fig. 4. Construction (here for $k = 4$) of a minimum-height ORD with $k + 1$ rows (above), whose minimum-row ORD (below) has 2 rows. (Not to scale vertically; distances between rows given on the left.)

of rows i and $i + 1$ in the minimum-height ORD D is exactly 2^{k-i}. We add an additional row of $2 \cdot k^{k-1}$ blocks on top of the drawing (the blue area in Fig. 4) that perfectly surround the k^{k-1} blocks on row 1. These blocks have top height 2^{k+1} and occupy the entire width of the row. Specifically, for every block on row 1 there is a corresponding block on row 0 with width 1 and bottom-height 0, and between two blocks of row 1 (and at the end) there is a block on row 0 with width $k-1$ and bottom-height 2^k. Therefore the total width of all blocks in the drawing is $2k^k$, and the height of the drawing is $2^{k+2} - 1$.

Lemma 2. *The ORD D constructed above has minimum height, and any other ORD D' with fewer rows is higher.*

Proof (sketch). It is easy to see that D has optimal height if the assignment of blocks to rows is fixed. Consider the first block b in D' that is assigned to a different row than in D. If b is in row i in D and in row $i+1$ in D', then D' is higher than D, as the height of D below row i is $\sum_{j=i}^{k} 2^{k-j} = 2^{k-i+1} - 1$ and the top height of b is already 2^{k-i+1}. If b is in row $i+1$ in D and in row i in D', then one of the blocks on row i cannot be placed below a truncated block of row $i-1$. Thus, the distance between row $i-1$ and i is 2^{k+2-i} instead of 2^{k+1-i}. Since the height of D below row i is only $2^{k-i+1} - 1$, D' must be higher than D.

Finally note that the total width of the blocks is exactly $2W$, and we can place the blocks on two rows. Thus, the minimum-row ORD has only 2 rows.

Theorem 3. *For ORDs, $\beta(n) = \Omega(\log n / \log \log n)$.*

3.2 Upper bound for minimum-height ORDs

We now prove that the bound in Theorem 3 is tight. That is, we show that the minimum-height ORD always has at most $O(\log n / \log\log n)$ times the number of rows of the minimum-row ORD. To this end we show that, given a minimum-height ORD D with "too many" rows, we can merge two rows into one. Since a block on one row influences where blocks on adjacent rows can be placed, local modification may not allow us to merge two rows, unless we can guarantee that the merged rows have enough flexibility to move blocks horizontally.

The flexibility we have to move blocks on a given row depends on how the blocks on adjacent rows are arranged. If all blocks on a row r are placed consecutively (they form one *megablock*), then on the next row there is enough flexibility to move blocks to merge (assuming the rows are not too full). However, if all blocks on r are regularly spaced, then on the next row there may not be enough space to move blocks at all. Hence, we move blocks on a sequence of consecutive rows until we create a row with a single megablock. This cannot always be done in a constant number of rows: we might need $\Omega(\log n / \log\log n)$ rows. To measure how close we are to a single megablock, we use an entropy-like function H on the free space between the blocks. We say a row r is δ-*dense* if the total width of the blocks on r is at most δW, where δ is the *density* of the row. We aim to show that H is always reduced by some term depending on δ.

Consider a set of k rows of the minimum-height ORD D, numbered row $1, \ldots, k$. Let δ_i be the density of row i. We first move and merge the blocks to obtain a *canonical placement* of the ORD. The blocks on row 1 are fixed. We then repeat the following operations until the ORD does not change anymore:

1. *Merge steps:* merge two consecutive megablocks b_1 and b_2 on the same row i into a larger megablock (see Fig. 5). A merge step is possible if and only if on rows $i-1$ and $i+1$, between b_1 and b_2, there is a large enough gap for the newly created megablock. There are *virtual* megablocks of width 0 on the left and right side of each row. That is, the leftmost megablock will jump into the leftmost available gap, and, if the row has more than one megablock, the rightmost megablock will jump into the rightmost available gap.
2. *Move steps:* move all blocks on row 2 to the left as much as possible. If a block starts hitting a block on row 3 below it, this block (and any other blocks on rows $i > 3$ further down) are moved along until a fixed block on a higher row is hit (possibly indirectly). The rightmost block on a row is moved to the right instead of to the left (unless there is only one megablock). After all blocks on row 2 are stuck, we fix the blocks on row 2 (only for this iteration of move steps) and repeat the process for the next row, and so on.

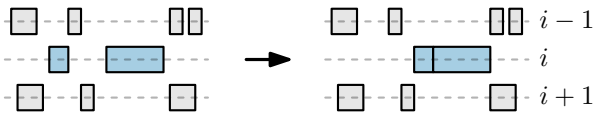


Fig. 5. Merge step: the two blue blocks are merged into a megablock.

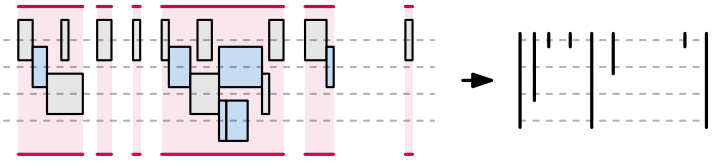


Fig. 6. Removing x-coordinates with blocks in them to obtain the free space partitioning. The rows in the drawing on the right represent $F_1, \ldots, F_4$.

As each merge step decreases the number of megablocks by one, we reach a canonical placement after a finite number of steps. In the remainder of this section we assume that we have a canonical placement of k rows. We say that a sequence of k rows is Δ-*dense* if the total width of all blocks is at most ΔW (typically, we use $\Delta < 1$). Clearly, for a set of Δ-dense rows, $\delta_i \leq \Delta$ for all i. In the following we assume w.l.o.g. that $W = 1$. Starting with the interval $[0, 1]$, we can obtain the *free space partitioning* of a set of rows by removing all x-coordinates occupied by any block (see Fig. 6). The free space partitioning is essentially a set of contiguous intervals of x-coordinates not used by any block. Note that the total length of the free space partitioning of a set of Δ-dense rows is at least $1 - \Delta$. Between two intervals of the free space partitioning there is an interval of x-coordinates occupied by blocks. We refer to such a set of blocks as a *separator*, and define the *index* of a separator S by the largest index of a row with a block in S (the leftmost and rightmost separator always have index k).

Lemma 3. *For a canonical placement, every internal separator S of index i contains a block for all rows j with $j \leq i$.*

We define the *i-th free space partitioning* F_i as the partitioning that includes only the separators with index at least i (see the right of Fig. 6), so F_i denotes gaps on row i. Further note that F_i is always a refinement of F_{i+1} by definition. We define the *entropy* [8] of F_i as $H_i = -\sum_j x_j \log x_j$, where x_j is the length of the j-th interval of F_i. We aim to show via the entropy that, if the number of rows k is large enough, then F_k contains at most two intervals.

Lemma 4. *For a canonical placement of a Δ-dense set of rows ($\Delta \leq 1/2$) with width $W = 1$, either F_{i+1} consists of at most two intervals, or $H_i - H_{i+1} \geq (1 - \Delta) \log((1 - \Delta)/(2\delta_{i+1}))$, where δ_{i+1} is the density of row $i + 1$.*

Proof. Consider an interval I of length y in F_{i+1}, where F_{i+1} has more than two intervals. Since F_i is a refinement of F_{i+1}, I is covered by r intervals with lengths $x_1, \ldots, x_r$ in F_i (see Fig. 7). Because the drawing is in canonical placement, $x_1, \ldots, x_r$ each must be smaller than the sum of the widths of the two megablocks in the separators surrounding I in F_{i+1}, which exist due to Lemma 3. (Otherwise these megablocks could have merged into that gap.) We denote the sum of the widths of the megablocks around I by z.

For this interval, the contribution to H_{i+1} is $-y \log y$, and the contribution to H_i is $\sum_j -x_j \log x_j$. Since $x_j < z$ for all j and $\sum_j x_j = y$, the contribution

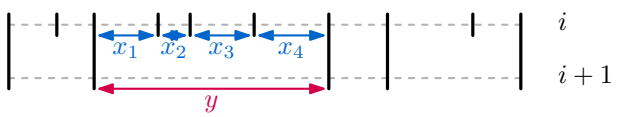


Fig. 7. Sketch of the free space partitionings F_i and F_{i+1}.

to H_i is at least $-y \log z$. Thus, the contribution to the difference $H_i - H_{i+1}$ in this interval is at least $y \log(y/z)$.

Next, we sum up these differences over all intervals of F_{i+1}. Let Z be the sum of all values z, and let Y be the sum of all values y. Y is the total length of the free space, which is at least $1 - \Delta$, while Z counts every block on row $i+1$ at most twice, thus $Z \leq 2\delta_{i+1}$. Using the log sum inequality we obtain that $H_i - H_{i+1} \geq Y \log Y/Z$. The claim follows from the bounds on Y and Z.

The next step is to use Lemma 4 to show that we can merge two rows without increasing height if we have sufficiently many rows. We first establish the conditions under which we can merge two consecutive rows.

Lemma 5. *Let D be an ORD with width W, 3 rows, and blocks of total width at most W. Then an ORD D' with only two rows exists which is not higher than D.*

Lemma 6. *Consider a Δ-dense set R of $2k+7$ consecutive rows of an ORD D with n blocks where $\Delta \leq 1/5$. If $(k/2)^{k/2} > 2n$, then we can obtain another ORD D' for which all rows not in R and the first and last row of R are the same as in D, D' has one fewer rows, and the height of D' is at most the height of D.*

Proof (sketch). First, we compute a canonical placement for the first $k+1$ rows, and, rotated by 180 degrees, also for the last $k+1$ rows (see Fig. 8). Using the log sum inequality, the entropy of row 1 is bounded by $H_1 \leq (1-\Delta)\log(n/(1-\Delta)) \leq \log(2n)$, since $\Delta \leq 1/2$. By applying Lemma 4 repeatedly, we obtain that both F_{k+1} and F_{k+7} have only two intervals. Place the blocks on row $k+2$ together (as one megablock) as far to the left as possible; note that there is at most Δ free space to the left of this megablock. Similarly, place the blocks on row $k+6$ as far to the right as possible. For rows $k+3$ to $k+5$, the interval between $[2\Delta, 1-2\Delta]$ is free. Since the remaining width is $1-4\Delta$, and the total width of the blocks on these 3 rows is at most Δ, we can apply Lemma 5 to the restricted interval $[2\Delta, 1-2\Delta]$ if $\Delta \leq 1/5$. This reduces the number of rows without increasing the height and thus completes the proof.

Lemma 7. *The minimum-height ORD D of an instance with n blocks has at most $O(r \log n / \log\log n)$ rows, where r is the number of rows of the minimum-row ORD.*

Proof. Assume w.l.o.g. that $W = 1$. Then the total width of all blocks is at most r. Now let k be the smallest integer such that $(k/2)^{k/2} > 2n$. Assume for the sake of contradiction that D has more than Crk rows for some large constant

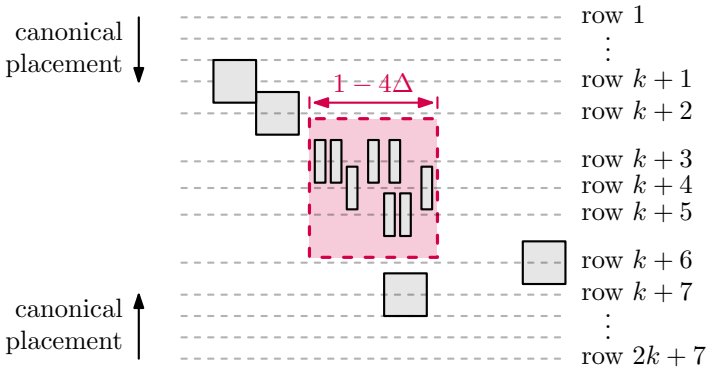


Fig. 8. Sketch of the proof of Lemma 6.

C. We partition the rows of D into consecutive groups of $(2k+7)$ rows. If none of these groups are Δ-dense for $\Delta = 1/5$, then we get that $r(Ck/(2k+7))/5 < r$ or $Ck/(10k+35) < 1$. We can easily choose C large enough (e.g. $C > 45$) such that this does not hold. Thus there must exist a Δ-dense group of $(2k+7)$ rows for $\Delta = 1/5$. We then apply Lemma 6 to obtain a contradiction. Thus, D has at most $O(rk)$ rows. To complete the proof, observe that $k = O(\log n / \log\log n)$.

Theorem 4. *For ORDs,* $\beta(n) = O(\log n / \log\log n)$.

References

1. Coffman, E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: Survey and classification. In: Handbook of Combinatorial Optimization, pp. 455–531. Springer (2013)
2. Hirschberg, D.S., Larmore, L.L.: The least weight subsequence problem. SIAM Journal on Computing **16**(4), 628–638 (1987)
3. Hofri, M.: Two-dimensional packing: Expected performance of simple level algorithms. Information and Control **45**, 1–17 (1980)
4. Knuth, D.E., Plass, M.F.: Breaking paragraphs into lines. Software—Practice and Experience **11**, 1119–1184 (1981)
5. Korte, B., Vygen, J.: Bin-Packing. In: Combinatorial Optimization, pp. 426–441. Springer (2005)
6. Kosaraju, S.R.: An optimal RAM implementation of catenable min double-ended queues. In: SODA. pp. 195–203 (1994)
7. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research **141**, 241–252 (2002)
8. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**, 379–423 (1948)
9. Sonke, W., Verbeek, K., Meulemans, W., Verbeek, E., Speckmann, B.: Optimal algorithms for compact linear layouts. In: Proc. 11th IEEE Pacific Visualization Symposium (PacificVis). pp. 1–10 (2018)
10. Wilber, R.: The concave least-weight subsequence problem revisited. Journal of Algorithms **9**(3), 418–425 (1988)