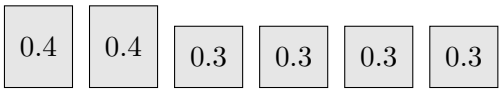
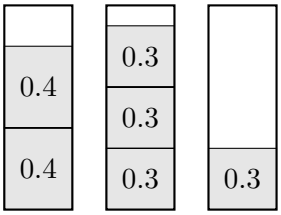
Bin Packing

1 Introduction

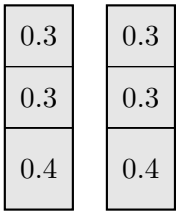
In the classical bin packing problem, we are given a set I of items. Each item $i \in I$ has a size $s(i) \in (0, 1]$ associated with it. Our goal is to partition I into the minimum number of bins, such that the sum of sizes of items in each bin is at most 1. See Fig. 1 for an example. The classical bin packing problem and its generalizations have diverse applications in computer science and operations research, like packing goods into trucks, allocating jobs to servers, allocating memory in computers [2], or assigning advertisements to station breaks in television programming.



(a) A set of six items: two items have size 0.4 and four items have size 0.3.



(b) A packing of the items into 3 bins.



(c) A packing of the items into 2 bins.

Figure 1: An example of classical bin packing. We want to minimize the number of bins, so the packing into 2 bins is better than the packing into 3 bins.

First-Fit Decreasing (FFD) is a popular algorithm for classical bin packing that can pack items I into at most $(11/9)\,\mathrm{opt}(I) + 4$ bins in $O(|I| \log |I|)$ time, where $\mathrm{opt}(I)$ is the minimum number of bins needed to pack I. Lueker and Vega [3]'s algorithm accepts a parameter $\varepsilon > 0$ and packs the items into $(1 + \varepsilon)\,\mathrm{opt}(I) + O(1/\varepsilon^2)$ bins in $O(|I \log |I| + 1/\varepsilon^2)$ time.

Let's now see a simple result on 1D bin packing.

Theorem 1. *In the classical bin packing problem, for any set I of items, we have*

$$\sum_{i \in I} s(i) \le \mathrm{opt}(I) < \left\lceil 2 \sum_{i \in I} s(i) \right\rceil.$$

Proof. For any set $X \subseteq I$ of items, let $s(X) := \sum_{i \in X} s(i)$ be the total size of the items in X. In an optimal packing, let B_j be the items in the j^{th} bin. Then

$$s(I) = \sum_{j=1}^{\mathrm{opt}(I)} s(B_j) \le \sum_{j=1}^{\mathrm{opt}(I)} 1 = \mathrm{opt}(I).$$

The first-fit algorithm packs each item into the first bin that it can fit in. Suppose the algorithm requires m bins, and let C_j be the items in the j^{th} bin. There cannot be two bins $j < k$ for which $s(C_j) \leq 1/2$ and $s(C_k) \leq 1/2$, otherwise the items C_k could have fit into C_j. Thus, at least $m - 1$ bins are more than half full. Hence,

$$s(I) = \sum_{j=1}^{m} s(C_j) > (m-1)\frac{1}{2}.$$

Thus, $\text{opt}(I) \leq m < 2s(I) + 1$, so $\text{opt}(I) \leq \lceil 2s(I) \rceil$. □

In the 2-dimensional geometric bin packing problem (abbreviated as 2D GBP), we are given a set I of n rectangular items and an infinite supply of identical rectangular bins. Our task is to pack the rectangles into the minimum number of bins such that in each bin, the items don't overlap. See Fig. 2 for an example.

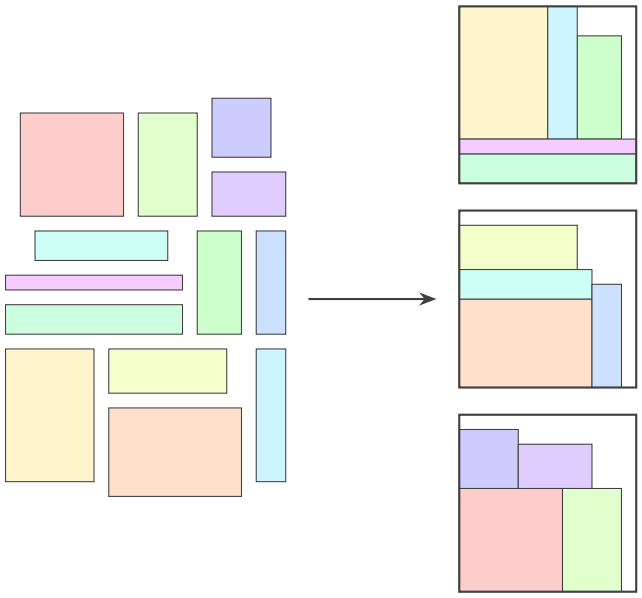


Figure 2: Packing 13 rectangles into 3 bins (without rotation).

There are two commonly-studied versions of 2D GBP. In the non-rotational version, rotating the items is forbidden. In the rotational version, the items can be rotated by 90°. In both versions, the items and bins are oriented parallel to the coordinate axes. 2D GBP finds applications in the wood-cutting, metal-cutting, paper and cloth industries, where rectangular pieces need to be cut out of standard-sized sheets, and item rotations are usually allowed. Non-rotational 2D GBP can be used for placing advertisements on web pages and newspapers.

For 2D GBP (both rotational and non-rotational versions), Bansal and Khan's algorithm [1] packs the items into $(\alpha + \varepsilon)\,\text{opt}(I) + O(1)$ bins, where $\alpha = 1 + \ln(1.5) \approx 1.4055$.

References

[1] Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *SODA*, pages 13–25, 2014. doi:10.1137/1.9781611973402.2.

[2] Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: Survey and classification. In *Handbook of Combinatorial Optimization*. Springer, 2013. doi:10.1007/978-1-4419-7997-1_35.

[3] W Fernandez De La Vega and George S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981. doi:10.1007/BF02579456.