The Bin Packing Problem

We consider packing problems of one dimension, though there is no conceptual difficulty extending the problem to *p* dimensions. In the one-dimensional problem objects have a single dimension (cost, time, size, weight, or any number of other measures). Problems of higher dimension have objects with more measures under consideration (cost and weight, length, width, and depth, etc.). The object is to pack a set of items into as few bins as possible. As the number of dimensions (measures) under consideration increases, the complexity increases tremendously, so we limit ourselves in this discussion to one dimension.

A formal definition of the bin packing (BP) problem follows.

Definition (Bin Packing Problem). Given a list of objects and their weights, and a collection of bins of fixed size, find the smallest number of bins so that all of the objects are assigned to a bin.

One dimensional bin packing is an NP-hard combinatorial optimization problem, based on the partition problem. Finding the optimal solution is known to be exponentially difficult. One solution methodology is to formulate the problem as an integer programming problem. The run-time performance of traditional exact methods, such as branch and bound, degrade as the problem size increases, however. In order to handle larger problems, heuristic methods have been developed. Some of the more popular algorithms are given in the following discussion.

> **Next Fit Heuristic (BP-NF).** Place the items in the order in which they arrive. Place the next item into the current bin if it fits. If it does not, close that bin and start a new bin.

Example. Given the set of items $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ and bins of size 10, pack the items into as few bins as possible using Next Fit.

The results of using the Next fit algorithm are shown in the figure below.

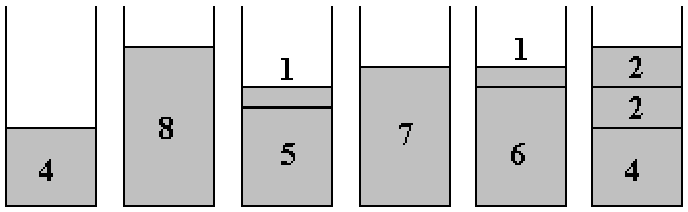

Figure 1 - Packing under Next Fit

A total of six bins are required to pack the items under Next Fit (compared to a best packing that requires 4 bins).

This algorithm is clearly wasteful of bin space, since the bins we close may not be very full (such as the first bin in the example). However, it does not require memory of any bin except the current one. This might be important if memory costs are a consideration, or if the staging area is limited (e.g., it can hold only one bin at a time).

If memory or staging is not an issue, we should be able to improve the performance of the algorithm by considering previous bins that might not be full. In effect, we expect that if we pack individual bins better, the overall solution will be better. Formalizing this idea leads to the First Fit algorithm. This algorithm is explained below.

> **First Fit Heuristic (BP-FF).** Place the items in the order in which they arrive. Place the next item into the lowest numbered bin in which it fits. If it does not fit into any open bin, start a new bin.

Example. Given the set of items *S* = {4, 8, 5, 1, 7, 6, 1, 4, 2, 2}
and bins of size 10, pack the items into as few bins as possible using First Fit.

The solution is shown here.

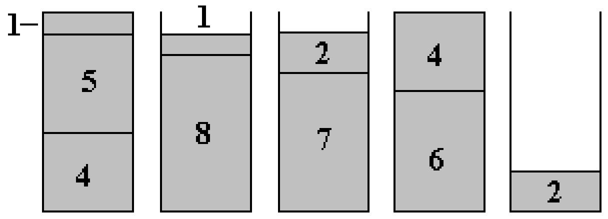

Figure 2 - Packing under First Fit

This algorithm requires memory of the available space in previous bins, and that these bins be available for further packing. Of course, there are variations to this idea such as limiting the memory or staging to *M* bins or looking ahead *d* items in the steam.

We might try to extend the idea in the First Fit algorithm by placing items in the best possible bin, that is, in the bin that will be filled the most by placing the item in it. In many cases we would fill the bin exactly. The general idea is to obtain the best global packing by getting the best local packing of individual bins. The algorithm is stated below.

> **Best Fit Heuristic (BP-BF).** Place the items in the order in which they arrive. Place the next item into that bin which will leave the least room left over after the item is placed in the bin. If it does not fit in any bin, start a new bin.

Example. Given the set of items *S* = {4, 8, 5, 1, 7, 6, 1, 4, 2, 2} and bins of size 10, pack the items into as few bins as possible using Best Fit.

This packing is shown in Figure 3.

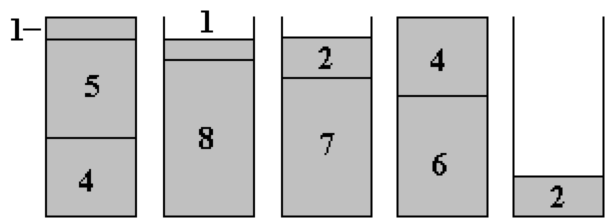

Figure 3 - Packing under Best Fit

In this example it turns out that the packings for BP-FF and BP-BF are the same, though generally some improvement in performance is observed with Best Fit.

A variation on Best Fit is Worst Fit.

> **Worst Fit Heuristic (BP-WF).** Place the items in the order in which they arrive. Place the next item into that bin which will leave the most room left over after the item is placed in the bin. If it does not fit in any bin, start a new bin.

Example. Given the set of items $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ and bins of size 10, pack the items into as few bins as possible using Worst Fit.

The solution to the problem using Worst Fit is shown in the figure.

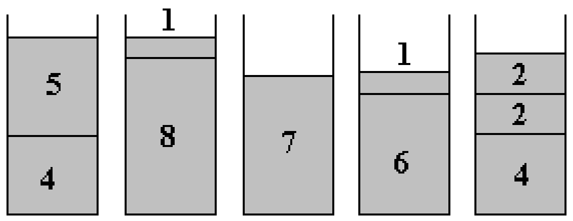

Figure 4 - Packing under Worst Fit

Worst Fit places the item into the most empty bin. This heuristic has the effect of spreading the slack (empty space) over the bins used. This algorithm might be useful if it is desirable to pack the bins with approximately the same weight or fill them with items of approximately the same value. A variation on this algorithm is the Almost Worst Fit heuristic, which places items into the second most empty bin. This algorithm is provably better than Worst Fit, and gives the best packing so far on our example data set.

Example. Given the set of items $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ and bins of size 10, pack the items into as few bins as possible using Almost Worst Fit.

This is the packing under Almost Worst Fit. In this case, the packing is optimal.

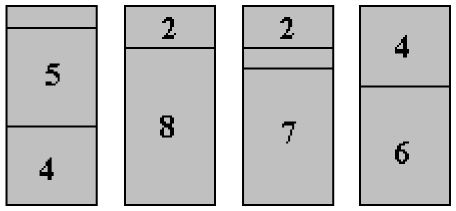

Figure 5 - Packing under Almost Worst Fit

If it is permissible to preprocess the list of items, significant improvements are possible for some of the heuristic algorithms. For example, if the items could be sorted before they are packed, a decreasing sort improves the performance of both the First Fit and Best Fit algorithms.

> **First Fit Decreasing Heuristic (BP-FFD).** Sort the items in decreasing order. Place the next item into the lowest numbered bin in which it fits. If it does not fit into any open bin, start a new bin.
>
> **Best Fit Decreasing Heuristic (BP-BFD).** Sort the items in decreasing order. Place the next item into that bin which will leave the least room left over after the item is placed in the bin. If it does not fit in any bin, start a new bin.

Example. Given the set of items *S* = {4, 8, 5, 1, 7, 6, 1, 4, 2, 2} and bins of size 10, pack the items into as few bins as possible using the BP-FFD and BP-BFD algorithms.

The items in sorted order are *S* = {8, 7, 6, 5, 4, 4, 2, 2, 1, 1}. Applying the First Fit algorithms gives the result below:

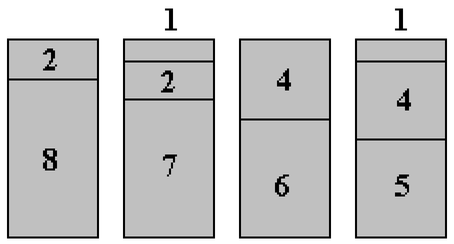

Figure 6 - Packing under First Fit Decreasing

The BP-BFD algorithm gives the same results. In both instances the packings improved when sorting was allowed.