Appears in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum (ed.), PWS Publishing, Boston (1996), 46-93

CHAPTER

2

APPROXIMATION ALGORITHMS FOR BIN PACKING: A SURVEY

E. G. Coffman, Jr. M. R. Garey D. S. Johnson

The classical one-dimensional bin packing problem has long served as a proving ground for new approaches to the analysis of approximation algorithms. In the early 1970's it was one of the first combinatorial optimization problems for which the idea of worst-case performance guarantees was investigated. It was also in this domain that the idea of proving lower bounds on the performance of online algorithms was first developed, and it is here that the probabilistic analysis of approximation algorithms has truly flowered. The chapter surveys the literature on worst-case and average-case behavior of approximation algorithms for one-dimensional bin packing, using each type of analysis to put the other in perspective.

2.1 INTRODUCTION

In the classical one-dimensional bin packing problem, we are given a sequence $L = (a_1, a_2, ..., a_n)$ of *items*, each with a *size* $s(a_i) \in (0, 1]$ and are asked to pack them into a minimum number of unit-capacity bins (i.e., partition them into a minimum number m of subsets $B_1, B_2, ..., B_m$ such that $\sum_{a_i \in B_j} s(a_i) \le 1$, $1 \le j \le m$).

This NP-hard problem has many potential real-world applications, from loading trucks subject to weight limitations to packing television commercials into station breaks [Bro71] to *stock-cutting* problems, where the bins correspond to standard lengths of some material, say cable, lumber, or paper, from which items must be cut.

Bin packing has also been of fundamental theoretical significance, serving as an early proving ground for many of the classical approaches to analyzing the performance of approximation algorithms. These include determining worst-case performance ratios (currently called *competitive ratios*) [Ull71] [JDU74], identifying lower bounds on the best possible online performance [Yao80], and analyzing average-case behavior [Sha77] [Lue82].

In this chapter, we survey the literature that has grown from these early papers. We concentrate on results for the basic problem defined above. A subsequent survey [CGJ97] will cover the wide variety of variants that it has spawned, including generalizations to higher dimensions, generalizations to include other constraints beyond bin capacity, and variants with different optimization criteria. The chapter can be viewed as a partial successor to two earlier surveys by the same authors, written in 1981 [GJ81] and 1984 [CGJ84]. As evidence of growth in the field, the number of references doubled between the first and second versions, and although this survey is restricted to the classical one-dimensional case, it has more references than [CGJ84] which covered the basic problem and all its variants.

We divide our coverage into two parts. The first (Section 2.2) covers worst-case results, while the second (Section 2.3) covers the average case. In both sections we consider the distinction between online and offline algorithms, where in the former, items arrive in some given order and must be assigned to bins as they arrive, without knowledge of the items yet to arrive. A brief final section (Section 2.4) offers general conclusions, and sketches the variety of variants of the basic problem that have been studied.

2.2 WORST-CASE ANALYSIS

In the case of bin packing, the standard metric for worst-case performance is the *asymptotic worst-case performance ratio.* For a given list L and algorithm A, let $A(L)$ be the number of bins used when algorithm A is applied to list L, let $\mathrm{OPT}(L)$ denote the optimum number of bins for a packing of L, and let $R_A(L) \equiv A(L)/\mathrm{OPT}(L)$. The *absolute* worst-case performance ratio R_A for algorithm A is defined to be

$$R_A \equiv \inf\{r \geq 1 : R_A(L) \leq r \text{ for all lists } L\}$$

The *asymptotic* worst-case performance ratio R_A^∞ is defined to be

$$R_A^\infty \equiv \inf\{r \geq 1 : \text{for some } N > 0, R_A(L) \leq r \text{ for all } L \text{ with } \mathrm{OPT}(L) \geq N\}$$

If in addition one restricts lists to those for which all items have sizes at most α, one can analogously define the *bounded-size* performance ratios, $R_A(\alpha)$ and $R_A^\infty(\alpha)$. Note that $R_A^\infty(1) = R_A^\infty$.

The remainder of this section is organized as follows. Sections 2.2.1 through 2.2.5 are all concerned with various types of online bin packing algorithms. Although for the general bin packing problem we assume that the entire list and its items' sizes are known before the packing begins, in many applications this may not be the case. A common situation is where the items arrive in some order, and must be assigned to a bin as soon as they arrive, without knowledge of the remaining items. This models situations in which items are physical objects, and there is no intermediate space to store them before placing them in bins. A bin packing algorithm that can construct its packings under this regime is called an *online* algorithm.

We begin in Sections 2.2.1 and 2.2.2 by introducing perhaps the two simplest and best known online algorithms, Next Fit and First Fit. Section 2.2.3 then considers generalizations and variations on First Fit, including Best Fit and the Almost Any Fit algorithms, all of which share with First Fit an asymptotic worst-case performance ratio of 1.7. Sections 2.2.4 and 2.2.5 examine the question of what is the best possible worst-case performance under the online constraint.

Section 2.2.4 does this first in the context of a second type of online constraint, that of *bounded space*. Now not only do the items arrive in an online fashion, but only a fixed number of partially-filled bins may be open to further items at any point in the packing process, and once a bin is closed it must remain so. This models situations in which bins are being exported once they are packed, and there is limited storage space for partially-filled ones. We present bounded-space online algorithms that perform as well as, and even slightly better than First Fit, but also note that the best possible behavior under this constraint is only marginally better than that obtained by First Fit.

Section 2.2.5 then considers arbitrary online algorithms, and whether more substantial improvements over First Fit can be obtained when the bounded-space constraint is removed. There is a limit on how much improvement can be obtained, as it can be shown that any online algorithm A must have $R_A^\infty > \beta$ for a constant $\beta > 1.5$. We discuss the best current upper and lower bounds known on β, and the algorithms on which the upper bounds depend (algorithms whose main motivation is this theoretical question, rather than potential applications in practice, where they are unlikely to perform as well as some of their simpler competitors).

Sections 2.2.6 through 2.2.9 cover offline algorithms. Section 2.2.6 covers what might be called *semi-online* algorithms, where the only relaxation of the online constraint is that each assignment of an item to a bin may be accompanied by a limited amount rearrangement of the other items in the current packing. Section 2.2.7 then covers what are perhaps the most famous of the offline algorithms, First and Best Fit Decreasing, with asymptotic worst-case ratios of $11/9 = 1.222...$ Section 2.2.8 covers other simple offline algorithms, including both ones that give up a little in worst-case performance for the sake of flexibility, simplicity, or speed, and ones that outperform First Fit Decreasing at a minor cost in increased algorithmic complexity. The best of these "simple" algorithms still has $R_A^\infty > 1.15$, but more complicated approximation schemes exist with asymptotic worst-case performance ratios approaching 1, and indeed there exist impractical but still polynomial-time bin packing algorithms with R_A^∞ (but not

$$L = \left(\frac{1}{2}, \frac{1}{2N}, \frac{1}{2}, \frac{1}{2N}, \ldots, \frac{1}{2}, \frac{1}{2N}\right)$$

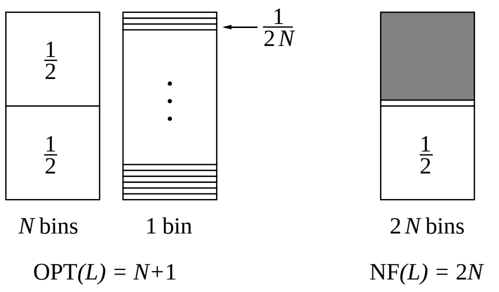

FIGURE 2.1

Worst-case examples for Next Fit.

R_A) equal to 1.0 exactly. These algorithms are discussed in Section 2.2.9.

Our discussion of worst-case results for the classical one-dimensional bin packing problem concludes in Section 2.2.10 by examining other questions that have been asked about the classic algorithms, besides the questions about R_A^∞ and $R_A^\infty(\alpha)$ on which we concentrate in Sections 2.2.1 through 2.2.9. Included are results about absolute worst-case performance ratios and bounds on anomalies, such as the situation in which deleting an item from a list causes the algorithm to use more bins.

2.2.1 NEXT FIT

Perhaps the simplest algorithm for the classical one-dimensional bin packing problem is Next Fit (NF), apparently first described under this name in [Joh73]. This is a bounded-space online algorithm in which the only partially-filled bin that is open is the most recent one to be started, i.e., the nonempty bin B_j in the current packing with the largest index j. (In this and subsequent discussions, we assume that bins are indexed $B_1, B_2, \ldots$ in the order in which they are created, i.e., receive their first items.) Let $level(B)$ be the sum of the sizes of the items in bin B. In packing item a_i, Next Fit tests whether $s(a_i) \leq 1 - level(B_j)$. If so, it places a_i in bin B_j, leaving that bin open. Otherwise, it closes bin B_j and places a_i in a new bin B_{j+1}, which now becomes the open bin.

This algorithm can be implemented to run in linear time, and it is not difficult to show that for all lists L, $\mathrm{NF}(L) \leq 2 \cdot \mathrm{OPT}(L) - 1$. Furthermore, there exist lists L with arbitrarily large values of $\mathrm{OPT}(L)$ such that $\mathrm{NF}(L) = 2 \cdot \mathrm{OPT}(L) - 1$, as illustrated in Figure 2.1. Thus we conclude that $R_{\mathrm{NF}}^\infty = 2$.

Note that the lists in the figure contain no items with $s(a) > 1/2$, so we can also conclude that $R_{\mathrm{NF}}^\infty(\alpha) = 2$ for all $\alpha \geq 1/2$. As α continues to decrease beyond this point, $R_{\mathrm{NF}}^\infty(\alpha)$ decreases in a continuous fashion, with the specific result being$R_{\mathrm{NF}}^{\infty}(\alpha) = 1/(1-\alpha)$ for $\alpha \le 1/2$ [Joh73].

2.2.2 FIRST FIT

As the worst-case examples of Figure 2.1 illustrate, Next Fit can be made to suffer because of the bounded-space constraint inherent in its definition. In Section 2.2.5, we shall consider the effect of partially relaxing this constraint to allow more than one but still a bounded number of open bins, as in the Next-K Fit algorithms of [Joh73]. For now we shall go directly to the historically more important First Fit algorithm, in which the restriction is removed entirely and we consider *all* partially-filled bins as possible destinations for the item to be packed. The particular rule followed is implicit in the algorithm's name: we place an item in the first (lowest indexed) bin into which it will fit, i.e., if there is any partially-filled bin B_j with $level(B_j) + s(a_i) \le 1$, we place a_i in the lowest-indexed bin having this property. Otherwise, we start a new bin with a_i as its first item. Note that in removing the bounded-space restriction, we forfeit the benefits of a linear running time enjoyed by Next Fit. We don't have to settle for the naive quadratic-time implementation, however, as it is possible to construct the First Fit packing in time $O(n \log n)$ using an appropriate data structure [Joh73]. (This is the best possible for a comparison-based implementation, since one can use the First Fit packing rule to sort [Joh73].)

First Fit manages to take good advantage of the wider range of destinations it considers, as the following result shows.

THEOREM 2.1 [GGJ76]. For all lists L, $\mathrm{FF}(L) \le \lceil (17/10) \cdot \mathrm{OPT}(L) \rceil$.

This is a slight tightening of what was essentially the first nontrivial bin packing result, proved by Ullman in 1971 [Ull71] [GGU71] with the slightly larger upper bound of $(17/10) \cdot \mathrm{OPT}(L) + 3$. The bound's additive term was reduced to 2 in the journal version of this paper [JDU74], and to 1 or less by the result of [GGJ76] highlighted above, although these improvements only reflected minor changes in the original proof. That proof has served as a model for many of the results that followed, so let us say a little bit about it.

The key idea is to use a *weighting* function $W : L \to \Re$, and to relate both $A(L)$ and $\mathrm{OPT}(L)$ to $W(L) \equiv \sum_{i=1}^{n} W(a_i)$ in such a way that the desired bound is implied. For Theorem 2.1, a static function suffices, i.e., one under which an item's weight depends only on its size. The version of W used in [GGJ76] is illustrated in Figure 2.2. Given this function one can use case analyses to prove two key lemmas: (i) If A is a set of items with $s(A) \equiv \sum_{a \in A} s(a) \le 1$, then $w(A) \equiv \sum_{a \in A} W(a) \le 17/10$, which implies $\mathrm{OPT}(L) \le (17/10) \cdot W(L)$, and (ii) $W(L) > \mathrm{FF}(L) - 1$. The theorem follows.

The upper bound of Theorem 2.1 is asymptotically tight, in that for arbitrarily large values of N one can construct lists L_N with $\mathrm{FF}(L_N) > (17/10) \cdot \mathrm{OPT}(L_N) - 2$ [JDU74]. These lists are fairly intricate, but simple examples that are almost as bad are readily constructed. Figure 2.3 illustrates a family of lists

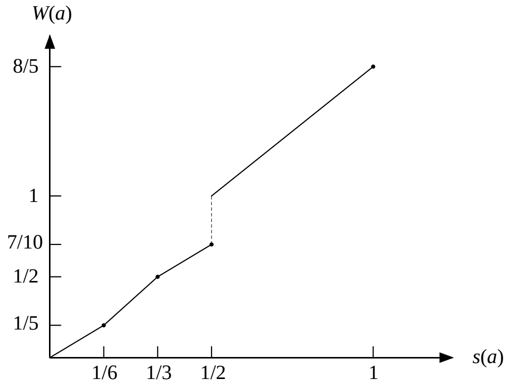

FIGURE 2.2

Weighting function W used in proof of Theorem 2.1.

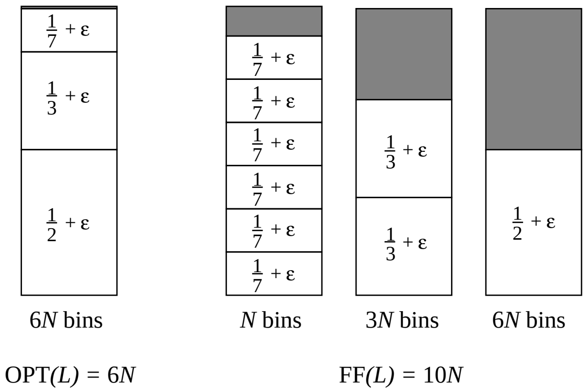

FIGURE 2.3

Bad lists for First Fit.

for which $\mathrm{FF}(L)/\mathrm{OPT}(L) = 5/3 = 1.666\ldots$ when the items are sorted in increasing order by size.

Note that still worse examples can be devised using the idea behind those in the figure. For instance, assuming ϵ is small enough, one could include an additional item of size $(1/43)+\epsilon$ in each bin of the optimal packing. Placing these at the beginning of the list will force First Fit to put them in an extra $N/42$ bins (assuming N is divisible by 42). Generalizing this idea, let $t_1 = 2$ and $t_{i+1} = (t_i)(t_i - 1) + 1$, $i > 1$. Then for any $k < \infty$ we can construct examples in which the optimal bins each contain k items, one each of size $(1/t_i)+\epsilon$, $1 \le i \le k$. When First Fit is given these items sorted in order of non-decreasing size, it will require roughly $\sum_{i=1}^{k} 1/(t_i - 1)$ bins. The best we can conclude from this, however, is that

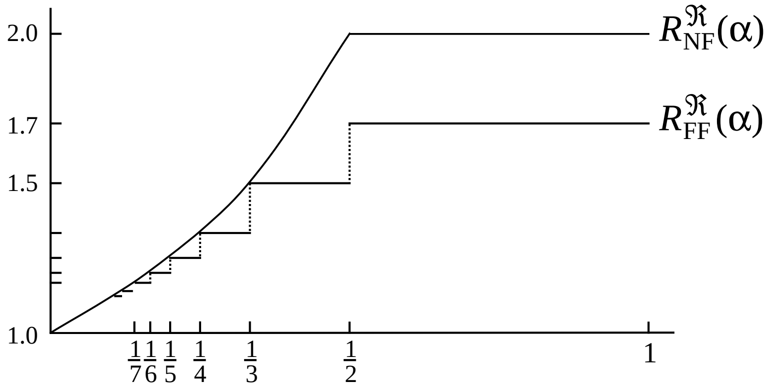

FIGURE 2.4

Comparison of $R^{\infty}_{\mathrm{NF}}(\alpha)$ and $R^{\infty}_{\mathrm{FF}}(\alpha)$ as functions of α.

$$R^{\infty}_{\mathrm{FF}} \geq T_{\infty} \equiv \sum_{i=1}^{\infty} \frac{1}{t_i - 1} = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \frac{1}{1805} + \cdots \approx 1.69103...$$

Although this simple scheme for worst case examples is thus insufficient to characterize the worst-case behavior of First Fit, we shall see below that there are other algorithms for which it and the constant T_{∞} are more relevant. Moreover, the scheme is correct in suggesting that for First Fit to behave at its worst, the instance to which it is applied must contain relatively large items. As with Next Fit, First Fit's worst-case behavior improves dramatically as the size of the largest item declines. Moreover, it maintains its advantage over Next Fit in such situations, although the size of its advantage depends on the precise value of α and shrinks with the size of the largest item.

THEOREM 2.2 [Joh73][JDU74]. Let $m \in \mathbf{Z}$ be such that $\frac{1}{m+1} < \alpha \leq \frac{1}{m}$.

A. For $m = 1$, $R^{\infty}_{\mathrm{FF}}(\alpha) = 17/10$

B. For $m \geq 2$, $R^{\infty}_{\mathrm{FF}}(\alpha) = 1 + 1/m$.

Figure 2.4 plots both $R^{\infty}_{\mathrm{NF}}(\alpha)$ and $R^{\infty}_{\mathrm{FF}}(\alpha)$ as functions of α. Note that although the value for NF approaches that for FF each time α approaches the reciprocal $1/t$ of an integer from above, it never catches it, as $R^{\infty}_{\mathrm{FF}}(\alpha)$ is a step function that changes value at precisely those points.

2.2.3 BEST FIT, WORST FIT, AND ALMOST ANY FIT ALGORITHMS

How crucial is the packing rule used by First Fit to the improved worst-case behavior outlined in the previous section? It turns out that other packing rules can do essentially as well. The most famous of these rules is the one used by the Best Fit (BF) algorithm. In this packing rule, which like the First Fit packing rule can be implemented to run in time $O(n \log n)$, item a_i is packed in the partially-filled bin B_j with the highest level $level(B_j) \le 1 - s(a_i)$, ties broken in favor of lower index. Best and First Fit can give strikingly different packings for individual lists. Examples are given in [Joh73] of lists L with arbitrarily large values of $\mathrm{OPT}(L)$ both such that $\mathrm{BF}(L) = (4/3) \cdot \mathrm{FF}(L)$ and such that $\mathrm{FF}(L) = (3/2) \cdot \mathrm{BF}(L)$. Nevertheless, all the results mentioned in the previous section for $R^{\infty}_{\mathrm{FF}}(\alpha)$ hold for $R^{\infty}_{\mathrm{BF}}(\alpha)$ as well [Joh73] [Joh74] [JDU74].

There are plausible packing rules for which these results do not hold, however. Consider the algorithm Worst Fit (WF), in which each item a_i is packed in the partially-filled bin with the *lowest* level (ties broken by index), assuming it fits, and otherwise starts a new bin. It is not difficult to see that $R^{\infty}_{\mathrm{WF}}(\alpha) = R^{\infty}_{\mathrm{NF}}(\alpha)$, $0 < \alpha \le 1$, so that in a worst-case sense, Worst Fit gets no value out of the fact that it never closes a bin.

Surprisingly, it takes only a slight modification to this algorithm to improve it dramatically. Consider the close variant Almost Worst Fit (AWF), in which a_i is placed in the partially-filled bin with the second lowest level (ties broken by index) unless there is only one bin into which it fits, in which case it goes in that bin. If a_i fits in no partially-filled bin, it of course starts a new bin. As shown in [Joh73] [Joh74], the worst-case behavior of Almost Worst Fit is just as good as that for First and Best Fit. Again, however, there can be significant differences on individual lists. Almost Worst Fit can be just as far off from Best Fit (in either direction) as was First Fit, and examples exist both such that $\mathrm{AWF}(L) = (5/4) \cdot \mathrm{FF}(L)$ and such that $\mathrm{FF}(L) = (9/8) \cdot \mathrm{AWF}(L)$ [Joh73].

More generally, let us say that an online bin packing algorithm is an *Any Fit* (AF) algorithm if it never starts a new bin unless the item to be packed does not fit in any partially-filled bin in the current packing, and that it is in addition an *Almost Any Fit* (AAF) algorithm if it never packs an item into a partially-filled bin with the lowest level unless there is more than one such bin or that bin is the only one that has enough room. Then we have the following result.

THEOREM 2.3 [Joh73][Joh74]. For all α, $0 < \alpha \le 1$,

A. If A is an AF algorithm, then $R^{\infty}_{\mathrm{FF}}(\alpha) \le R^{\infty}_{A}(\alpha) \le R^{\infty}_{\mathrm{NF}}(\alpha)$, and

B. If A is an AAF algorithm, then $R^{\infty}_{A}(\alpha) = R^{\infty}_{\mathrm{FF}}(\alpha)$.

Thus, if we want to obtain better worst-case behavior than that of First Fit, we shall either have to abandon the Any Fit constraint or the online restriction itself. In the next two sections we shall discuss what can be done while still obeying the online restriction.

2.2.4 BOUNDED-SPACE ONLINE ALGORITHMS

One disadvantage of the Any Fit constraint beside that implied by Theorem 2.3 is that under it no bin can ever be permanently closed to further items. Consequently no Any Fit algorithm can be used in situations where the *bounded-space* constraint described at the beginning of Section 2.2 holds, i.e., where at most K bins can remain open at any given time for some fixed K. In this section we consider what can be accomplished when space *is* bounded.

We have already seen one bounded-space online algorithm, Next Fit, and observed that it paid a substantial worst-case performance penalty in comparison to the unbounded-space algorithms First and Best Fit. Such a penalty is unavoidable if the bound K on the number of open bins is 1. When $k > 1$, four natural hybrids between Next Fit and First and Best Fit suggest themselves. Note that in constructing a bounded-space online algorithm we need to specify a *closing* rule in addition to a packing rule. First and Best Fit each suggest one rule of each type.

Assuming some open bin has room for the current item b, the First Fit packing rule places b in the lowest indexed open bin that has room for it and the Best Fit packing rule places b in the highest-level open bin that has room (ties broken in favor of lowest index). When there are already K open bins and none of them has room for b, some bin must be closed before a new bin can be opened to receive b. The First Fit closing rule closes the lowest indexed open bin, and the Best Fit closing rule closes the fullest open bin (ties broken in favor of lowest index).

The Next-K Fit algorithm (NF_K), introduced in [Joh73], combines the First Fit packing and closing rules. The K-Bounded Best Fit algorithm (BBF_K), introduced in [CJ91] combines the Best Fit packing and closing rules. The two remaining hybrids mix Best and First Fit rules and are denoted by ABF_K and AFB_K in [CJ91]. The first algorithm, ABF_K, uses the Best Fit packing rule and the First Fit closing rule and was analyzed by [Mao93a] under the name *Best-K Fit*. The second, AFB_K, uses the First Fit packing rule and the Best Fit closing rule. All four algorithms reduce to Next Fit when $K = 1$, and all can be implemented to run in linear time for any fixed K, as opposed to the $\Omega(n \log n)$ time needed by our implementations of First and Best Fit. The next theorem summarizes what is known about the asymptotic worst-case ratios for the algorithms when $K \geq 2$.

THEOREM 2.4 [CI89] [Mao93b] [Mao93a] [Zha94] [CJ91]. For $K \geq 2$,

- ***A.*** $R^\infty_{\mathrm{NF}_K} = R^\infty_{\mathrm{AFB}_K} = 1.7 + \frac{3}{10(K-1)}$
- ***B.*** $R^\infty_{\mathrm{ABF}_K} = 1.7 + \frac{3}{10K}$
- ***C.*** $R^\infty_{\mathrm{BBF}_K} = 1.7$

The lower bound examples for $R^\infty_{\mathrm{NF}_K}$ were discovered by Csirik and Imreh [CI89] and the matching upper bound was proved by Mao [Mao93b], who also proved the result for $R^\infty_{\mathrm{ABF}_K}$ [Mao93a]. The result for $R^\infty_{\mathrm{AFB}_K}$ was proved by Zhang

[Zha94], and that for $R^{\infty}_{\mathrm{BBF}_K}$ was proved by Csirik and Johnson [CJ91]. All the upper bound proofs rely on weighting function arguments similar to those used for FF in Theorem 2.1.

The surprise here is the result for BBF_K, which yields the same performance guarantee as do First and Best Fit as soon as $K \geq 2$, although as might be expected, the penalty for the space bound approaches 0 for the other three algorithms as $K \to \infty$. One could not have expected any of these algorithms to do *better* than First and Best Fit, for although they do not obey the Any Fit constraint, they do obey the bounded-space analog of that constraint, and the proof of Theorem 2.3 implies that any such algorithm must have $R^{\infty}_{A} \geq 1.7$. If we are to find an algorithm with $R^{\infty}_{A} < 1.7$, we will thus have to consider algorithms that start new bins even when the current item fits in one of the currently open bins.

The worst-case examples of Figures 2.1 and 2.3 suggest the following series of bounded-space algorithms, introduced by Lee and Lee in [LL85] as the *Harmonic*$_K$ algorithms (H_K). Divide the unit interval into K subintervals I_k, $1 \leq k \leq K$, where $I_k = (1/(k+1), 1/k]$, $1 \leq k < K$, and $I_K = (0, 1/K]$. An item a will be identified as having *type* k if $s(a) \in I_k$. Similarly, bins will be divided into K types, with bins of type k only receiving items of type k, and at most one bin of each type open at any time. Each item a is then packed as follows: Let k be the type of item a. If there is an open bin of type k and that bin has room for a, then place a in that bin. Otherwise, close the open bin of type k if one exists, and place a in a new (open) bin of type k. For K sufficiently large, these algorithms finally break the 1.7 barrier, with a limiting value that equals the constant that we encountered in Section 2.2.2.

THEOREM 2.5 [LL85]. $\lim\limits_{K \to \infty} R^{\infty}_{\mathrm{H}_K} = T_{\infty} = 1.69103..$

Indeed, as soon as $K \geq 7$, we have $R^{\infty}_{\mathrm{H}_K} \leq 1.695$. One can actually get performance guarantees less than 1.7 for K as small as 6, but this requires a variant on the original Harmonic scheme introduced by Woeginger [Woe93]. Woeginger's algorithms, which he calls the *Simplified Harmonic* algorithms (SH_K), rely on a more complicated interval structure, derived from the sequence of t_i's defined in Section 2.2.2. Table 2.1 summarizes what is known about the values of R^{∞}_{A} as a function of K for the various bounded-space algorithms discussed in this section. Tight bounds for H_K are not known for all values of K, so the table gives the best upper and lower bounds currently known. In this table, the upper bounds for all values of K except 4 and 5 are from [LL85]. The upper bounds for $K \in \{4, 5\}$ are due to van Vliet [Vli95] and are tight. The lower bounds for $K \geq 4$ are due independently to [CJ92] and to [Vli95] [Vli96].

Note that BBF_K provides the best guarantee for $K \in \{2, 3, 4\}$, BBF_K, H_K, and SH_K are tied for the best when $K = 5$, and thereafter SH_K is the best. Asymptotically, it uses only $O(\log\log K)$ as many open bins to obtain the same performance guarantee as H_K, although its margin over H_K for particular values of K is never more than about 0.3%, and rapidly declines to 0 as $K \to \infty$. Asymptotically, both are only about 0.5% better than FF, BF, and BBF_K. This

K	NF_K	ABF_K	BBF_K	$\mathrm{H}_K \geq$	$\mathrm{H}_K \leq$	SH_K
2	2.00000	1.85000	1.70000	2.00000	2.00000	2.00000
3	1.85000	1.80000	1.70000	1.75000	1.75000	1.75000
4	1.80000	1.77500	1.70000	1.71429	1.71429	1.72222
5	1.77500	1.76000	1.70000	1.70000	1.70000	1.70000
6	1.76000	1.75000	1.70000	1.70000	1.70000	1.69444
7	1.75000	1.74286	1.70000	1.69444	1.69444	1.69388
8	1.74286	1.73750	1.70000	1.69377	1.69388	1.69106
9	1.73750	1.73333	1.70000	1.69326	1.69345	1.69104
10	1.73333	1.73000	1.70000	1.69287	1.69312	1.69104
∞	1.70000	1.70000	1.70000	1.69103	1.69103	1.69103

Table 2.1: Values of R_A^∞ under fixed space bounds, rounded to five decimal places.

is not a major advantage, but it is all that is possible for bounded-space online algorithms in light of the following result of Lee and Lee.

THEOREM 2.6 [LL85]. If A is any bounded-space online bin packing algorithm, then $R_A^\infty \geq T_\infty = 1.69103..$

It should be noted that at present no online bounded-space algorithm A is know for which $R_A^\infty = T_\infty$. The worst-case ratios for the sequences of algorithms H_K and SH_K only approach this value in the limit. If one is willing to consider bounded-space algorithms that are only semi-online, however, there are algorithms whose worst-case ratios match the limiting value, as has recently been shown by Galambos and Woeginger [GW93b] and [Gro94]. The relaxation used by [GW93b] is to allow *repacking* of the current open bins, i.e., to allow us to take all the items out of the current open bins and reassign them before packing the current item. The conclusion of Theorem 2.6 continues to hold even if we allow repacking [GW93b], but Galambos and Woeginger present an "online with repacking" algorithm REP_3 that never uses more than three open bins and yet has $R_{\mathrm{REP}_3}^\infty = T_\infty$. Grove [Gro94] independently constructed an algorithm with the same behavior using an alternative notion he calls *lookahead*. In such an algorithm, one is given a fixed *warehouse size* W, and an item a_i need not be packed until one has looked at all items a_i through a_j, for $j > i$ such that $\sum_{h=i}^{j} s(a_h) \leq W$. Allowing lookahead does not allow us to escape from the constraints of Theorem 2.6 either, but if one allows sufficiently large (fixed) values of K and W, Grove's *Warehouse* algorithm can again guarantee an asymptotic worst-case ratio of T_∞.

As remarked above, however, $T_\infty = 1.691...$ is not that great an improvement over 1.7. If an online algorithm is to improve significantly over the worst-case behavior of First Fit, it must exploit something stronger than bounded-space repacking or lookahead, i.e., it must allow for the use of unbounded space. In light of Theorem 2.3, it must also be prepared to disobey the Any Fit constraint and start new bins even when the current item will fit in some already-started bin. We discuss such algorithms in the next section.

2.2.5 ARBITRARY ONLINE ALGORITHMS

The first unbounded-space algorithms to be proposed that did not obey the Any Fit constraint were the Group-X Fit (GXF) algorithms of Johnson [Joh73] [Joh74]. One gets different versions of the algorithm depending on the choice of X, which is an increasing sequence $0 = x_0 < x_1 < ... < x_p = 1$ of real numbers. At any given time, a partially-filled bin with gap g is viewed as having a *de facto* gap equal to $\max\{x_i \le g : 0 \le i < p\}$. This in effect partitions the bins into p groups. Items are then packed via Best Fit with respect to the de facto gaps. Note that, like our bounded-space algorithms, this algorithm can be implemented to run in linear time for any fixed X. Now, however, the only bins to be effectively closed are those with gaps smaller than x_1, which hence get de facto gaps of 0. Unfortunately, the main advantage of GXF seems to be its linear running time, as the algorithm does not avoid the worst-case examples for First Fit and so has $R^{\infty}_{\mathrm{G}X\mathrm{F}} > 1.7$. Johnson did show, however, that for $m = \lfloor 1/\alpha \rfloor \ge 2$, $R^{\infty}_{\mathrm{G}X\mathrm{F}}(\alpha) = R^{\infty}_{\mathrm{FF}}(\alpha)$ whenever $\{1/(m+1), 1\} \subseteq X$, and conjectured that the same equality held for $m = 1$ whenever $\{1/6, 1/3, 1/2\} \subseteq X$ [Joh73].

The first online algorithm with $R^{\infty}_{A} < T_{\infty}$ (and actually the first one to beat First Fit, since it preceded the Harmonic algorithm of [LL85]), was the Refined First Fit (RFF) algorithm of Yao [Yao80]. This algorithm classified items as to which of the intervals $(0, 1/3]$, $(1/3, 2/5]$, $(2/5, 1/2]$, and $(1/2, 1]$ their sizes fell, and classified bins according to a related scheme. Each item was then packed into a particular class of bins according to the First Fit rule, with the bin class being determined by the item class and, in the case of the items with sizes in $(1/3, 2/5]$, the number of such items previously encountered in the list. For this type of item, one in 6 are treated specially, the idea being to occasionally start a new bin with an item of this size in hopes of subsequently adding an item of size greater than 1/2 to that bin. This will allow the algorithm to defeat the worst-case examples for First Fit (and the Harmonic algorithms), while not opening the way for an adversary to do too much damage by providing many items with sizes in $(1/3, 2/5]$ but no later items in $(1/2, 1]$ that will pair up with them. Yao showed that $R^{\infty}_{\mathrm{RFF}} = 5/3 = 1.666...$

This paper set off something of a race, which has led to more and more elaborate variants. The first was the Refined Harmonic (RH$_K$) algorithms of [LL85], a hybrid of the Harmonic algorithms with Refined First Fit. RH$_K$ uses the partitioning scheme of HK$_{20}$, with the modification that the two size-intervals $(1/3, 1/2]$ and $(1/2, 1]$ are replaced by the four intervals $(1/3, y]$, $(y, 1/2]$, $(1/2, 1-y]$, and $(1-y, 1]$, where $y = 37/96$. Packing proceeds much as in a Harmonic algorithm, except now one attempts to pair items whose sizes are in the first of the new intervals with items whose sizes are in the third new interval, since such pairs can always fit in the same bin. As in Refined First Fit, one must hedge one's bet when doing this, and here only one in 7 of the items with sizes in the first interval get the special treatment. Lee and Lee showed that $R^{\infty}_{\mathrm{RH}_{20}} \le 373/228 = 1.6359...$, with little to be gained by increasing K further.

Next came Ramanan, Brown, Lee, and Lee [RBL89], with what they called the Modified Harmonic (MH$_K$) algorithms, which added the possibility of packing still smaller items with items of size in $(1/2, 1/2+y]$, and a consequently more complicated algorithmic structure (as well as a different value for y, in this case $y = 265/684$). The details are too complex to be gone into here, but Ramanan et al. were able to show that $1.6156146 < R^{\infty}_{\mathrm{MH}_{38}} \le 1.(615)^* \equiv 1.615615....$ Shortly after drafts of [RBL89] began to circulate, Hu and Kahng [HK88] used somewhat similar principles to construct an (unnamed) variant for which they claimed $R^{\infty}_{A} \approx 1.6067$. Ramanan et al. themselves sketched further variants which they thought might take the asymptotic worst-case ratio down to as little as 1.59, but proved that the basic approach could never yield $R^{\infty}_{A} < 1.(583)^*$.

The current champion, Richey's Harmonic+1 algorithm [Ric91], uses somewhat different principles, but essentially attains the limit claimed by Ramanan et al., at least to the first three decimal places. Although it runs in linear time like its predecessors, it is substantially more complicated, with a significantly more complex interval structure involving over 70 intervals and with a packing rule that itself depends on an almost full-page table of constants. Richey shows that for this algorithm, $1.5874 \le R^{\infty}_{A} \le 1.588720$, and that the lower bound will hold for any variant that takes roughly the same approach.

Whether some other approach might do better remains an open question. Given the complicated nature of Harmonic+1 and the fact (as we shall see in Section 2.2.2) that all algorithms that do better than First Fit in the worst case seem to do much worse in the average case, this is perhaps not an open question than needs an answer. We can, however, place bounds on how much improvement might be possible, analogous to the bound on bounded-space online algorithms of Theorem 2.6.

The first such bound was proved by Yao in [Yao80], and actually preceded the abovementioned bounded space result. Yao showed that no online algorithm could have $R^{\infty}_{A} < 1.5$, using an adversary argument that required only three item sizes: $1/2+\epsilon$, $1/3+\epsilon$, and $1/7+\epsilon$. Note that these item sizes are simply $1/t_i+\epsilon$ for $1 \le i \le 3$ and the t_i used in the definition of T_∞ given in Section 2.2.2. Shortly thereafter Brown and Liang independently generalized this approach to use items of size $1/2+\epsilon$, $1/3+\epsilon$, $1/7+\epsilon$, $1/43+\epsilon$, and $1/1807+\epsilon$, i.e., $1/t_i+\epsilon$ for $1 \le i \le 5$, and proved that no online algorithm could have $R^{\infty}_{A} < 1.536346...$ [Bro79] [Lia80] . A simplified version of this proof has recently been presented in [GF93]. Because of the rapid growth in the values t_i, attempts to improve this bound by allowing the adversary additional item sizes $1/t_i+\epsilon$ for $i > 5$ do not seem likely to yield much increase in the bound. More careful analysis using just the values for $i \le 5$ can make a difference however, as shown by van Vliet [Vli95] [Vli96]. Building on an approach first proposed by Galambos [Gal86], van Vliet constructs a linear program whose solution specifies the best possible way of exploiting these item sizes, and derives the currently best lower bound known:

THEOREM 2.7 [Vli95][Vli96]. For any online algorithm A, $R^{\infty}_{A} \ge 1.540$.

This sort of lower bound analysis can be extended to questions about $R^{\infty}_{A}(\alpha)$, and Table 2.2 summarizes the current best bounds known for $\alpha = 1/m$, $1 \le m \le 5$, comparing them with the best values known for $R^{\infty}_{A}(\alpha)$ with A online, with the limiting value of $R^{\infty}_{\mathrm{H}_K}(\alpha)$ (as $K \to \infty$), and with $R^{\infty}_{\mathrm{FF}}(\alpha)$. The lower bounds are due again to [Vli95] [Vli96], improving on earlier results of [Gal86]. Note that by the time $\alpha = 1/5$, even First Fit is providing guarantees within 1% of best possible.

m	Lower Bound	Best Known	$R^{\infty}_{\mathrm{H}_K}(1/m)$	$R^{\infty}_{\mathrm{FF}}(1/m)$
1	1.540...	1.588...	1.691...	1.700...
2	1.389...	1.423...	1.423...	1.500...
3	1.291...	1.302...	1.302...	1.333...
4	1.229...	1.234...	1.234...	1.250...
5	1.188...	1.191...	1.191...	1.200...

Table 2.2: Best possible and actual values of $R^{\infty}_{A}(1/m)$ for online algorithms.

The proofs of the above lower bound results all assume that the online algorithm is deterministic. It appears, however that this is not necessarily a required assumption. Chandra [Cha92] has shown that even if we allow an online algorithm to base its assignments on random coin flips, and use the expected length of the resulting packing as our metric, there still exist lists that yield ratios of $E[A(L)]/\mathrm{OPT}(L)$ approaching 1.536... (Chandra's proof technique seems general enough to extend as well to the other lower bounds summarized above).

2.2.6 SEMI-ONLINE ALGORITHMS

As we saw at the end of Section 2.2.4 for bounded-space online algorithms, relaxing the online restriction slightly by allowing bounded repacking or lookahead can yield better algorithms. However, whereas in the case of bounded space such relaxations only allowed us to attain rather than beat the online lower bound, the situation is quite different in the general online case, at least for sufficiently powerful notions of bounded repacking.

Note first that if no limit on repacking is imposed, then one can do as well as the best offline algorithm, simply by repacking everything according to that algorithm each time an item arrives. This, however, would multiply the overall running time by a factor of n, and introduce at least linear-time delays each time a new item arrives. (By delaying the repackings appropriately, one can reduce the amortized delay per new item to $T(n)\log n/n$, where $T(n)$ is the offline algorithm's running time, but the worst-case delay would remain $T(n)$ [IL94].) In applications that still retain something of an online flavor, one would presumably need stronger restrictions on how much repacking is allowed, for instance allowing only constant or $O(\log n)$ time per item in a worst-case sense. Even under such restrictions, however, major improvements can be obtained over the 1.540 lower bound on R^{∞}_{A} for pure online algorithms.

The first authors to observe this were Gambosi et al. [GPT90]. They designed two algorithms that beat the bound. The first yielded $R^{\infty}_{A} \leq 1.5$ using only constant time per item. In the worst case $\Omega(n)$ items might have to be moved while accommodating a single new item, but with the aid of appropriate data structures Gambosi et al. could treat large collections of small items as a group and move them all at once in constant time. In a physical bin packing situation, this might correspond to keeping collections of small items in boxes and moving entire boxes from one bin to another. In this algorithm, which was based on a simple classification of items into four types by size, no more than 3 group or single-item moves are ever required when a new item has to be assigned.

The second algorithm of [GPT90] had six types of items, required $\Theta(\log n)$ time but at most 7 group/item moves per new item, and yielded $R_A^\infty \leq 4/3 = 1.333...$. More recently, Ivković and Lloyd [IL93] have devised an algorithm with $R_A^\infty \leq 5/4 = 1.25$, although for this they need as many as $O(\log n)$ group/item movements as well as $O(\log n)$ time per new item. They also have significantly more complicated packing/repacking rules, although part of this is because their algorithm can also handle the dynamic bin packing problem where items can depart as well as arrive. (It has the same bound on asymptotic worst-case performance in this more generalized situation.)

This is currently the best worst-case behavior currently known for such semi-online algorithms. If we want provably better asymptotic worst-case ratios, we must turn to algorithms that are offline and hence have access to all the items before any of them need to be assigned to bins. The next section covers the most famous of these, ones that were investigated long before many of the above questions about the online case were even considered.

2.2.7 FIRST FIT DECREASING AND BEST FIT DECREASING

Looking at the instances in Figure 2.3 that make First Fit misbehave, it is clear that there are dangers in lists of items sorted by increasing size. Thus a natural idea for improving on First Fit once the online restriction is removed would be to sort the list in some other way before applying the First Fit packing rule. In the *First Fit Decreasing* (FFD) algorithm, the items are first sorted in order of non-increasing size, and then the First Fit packing rule is applied. The algorithm *Best Fit Decreasing* (BFD) is defined analogously, using the Best Fit packing rule. The improvement over First and Best Fit is dramatic.

THEOREM 2.8 [Joh73]. $R_{\mathrm{FFD}}^\infty = R_{\mathrm{BFD}}^\infty = 11/9 = 1.222...$

Examples of instances that provide the lower bound for this result are given in Figure 2.5, and first appeared in [GGU71]. The upper bound in [Joh73] was more precisely $\mathrm{FFD}(L) \leq (11/9) \cdot \mathrm{OPT}(L) + 4$ for all lists L. The proof of this was much more complicated than that for Theorem 2.1, with a weighting function that depended not only on item sizes but locations in the packing, and 70 pages of case analysis. Subsequently, Baker devised a somewhat simpler proof and reduced the additive constant from 4 to 3 [Bak83], and Yue has claimed a much simpler proof and a reduction of the additive constant to 1 [Yue91].

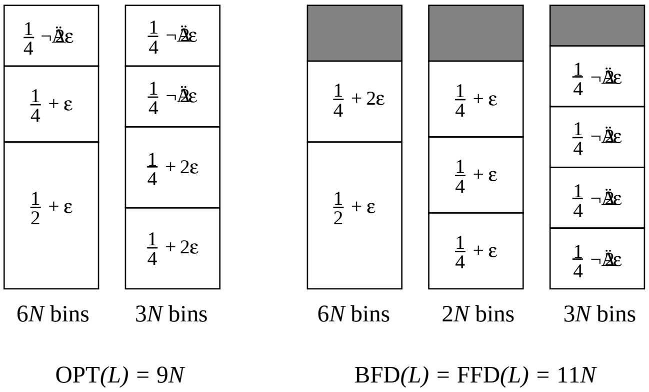

FIGURE 2.5

Worst-case examples for First and Best Fit Decreasing.

The analogous results for Best Fit Decreasing follow from the fact that for all lists L with no item smaller than $1/6$, $\mathrm{BFD}(L) \le \mathrm{FFD}(L)$ [Joh73] [JDU74]. This is enough, since any list in which an item of size less than $1/6$ starts a bin under BFD must have a BFD packing in which all bins except the last have level exceeding $5/6$. Hence for such lists $\mathrm{BFD}(L) \le (6/5) \cdot \mathrm{OPT}(L) + 1$. BFD can produce worse packings than FFD, however, if smaller items are allowed. Lists L are shown in [Joh73] [JDU74] for which $\mathrm{BFD}(L) = (10/9) \cdot \mathrm{FFD}(L)$. BFD can produce *better* packings than FFD as well, so long as items as small as $1/5 - \epsilon$ are allowed, and lists L of this sort for which $\mathrm{FFD}(L) = (11/10) \cdot \mathrm{BFD}(L)$ are shown in the same reference.

As with the earlier algorithms, the worst-case ratios for First and Best Fit Decreasing improve as the maximum item size α declines, although now the dependence on α is more complicated than it was for the earlier algorithms. The examples in Figure 2.5 imply that $R^{\infty}_{\mathrm{FFD}}(\alpha) = 11/9$ for $1/2 < \alpha \le 1$. In [Joh73], it was shown that

$$R^{\infty}_{\mathrm{FFD}}(\alpha) = \begin{cases} \frac{71}{60}, & \frac{8}{29} < \alpha \le \frac{1}{2} \\ \frac{7}{6}, & \frac{1}{4} < \alpha \le \frac{8}{29} \\ \frac{23}{20}, & \frac{1}{5} < \alpha \le \frac{1}{4} \end{cases}$$

It was also conjectured that for integers $m \ge 4$,

$$R^{\infty}_{\mathrm{FFD}}(1/m) = F_m \equiv 1 + \frac{1}{m+2} - \frac{2}{m(m+1)(m+2)}$$

.

This conjecture turns out to hold only when m is even. When m is odd, Csirik [Csi93] subsequently showed that

$$R^{\infty}_{\mathrm{FFD}}(1/m) = G_m \equiv 1 + \frac{1}{m+2} - \frac{1}{m(m+1)(m+2)}$$

.
The question of what the situation is for general values of α was answered by Xu in [Xu93], which showed that when m is even, $R^{\infty}_{\mathrm{FFD}}(\alpha)$ equals F_m throughout the interval $(1/(m+1), 1/m]$, but when m is odd, there is a d_m, $1/(m+1) < d_m < 1/m$, such that $R^{\infty}_{\mathrm{FFD}}(\alpha)$ equals G_m only in the interval $(d_m, 1/m]$ while equaling F_m in the interval $(1/(m+1), d_m]$. (The precise value of d_m is $(m+1)^2/(m^3+3m^2+m+1)$.)

Note that both F_m and G_m represent significant improvements over the results for First Fit cited above, where $R^{\infty}_{\mathrm{FF}}(1/m) = 1+(1/m)$.

2.2.8 OTHER SIMPLE OFFLINE ALGORITHMS

The choice of a packing rule could make a big difference in the case of online algorithms. One might wonder whether it is so crucial in the context of lists sorted by decreasing size. In the case of the NF versus FF comparison, use of decreasing lists actually amplifies the difference in worst-case behavior. Consider Next Fit Decreasing (NFD), the algorithm that puts the items in decreasing order by size and then applies Next Fit. As proved in [BC81], we have $R^{\infty}_{\mathrm{NFD}} = T_{\infty} = 1.69103\ldots$ Note that this limit exceeds 11/9 by more than $R^{\infty}_{\mathrm{NF}} - R^{\infty}_{\mathrm{FF}} = 3/10$.

More typically, however, the difference between packing rules shrinks when we consider only lists ordered by decreasing item size. For Any Fit packing rules the maximum difference shrinks from the .3 we saw in Section 2.2.3 to .0277..., as a consequence of the following.

THEOREM 2.9 [Joh73][Joh74]. For any algorithm A that sorts the items by decreasing size and then applies an Any Fit packing rule,

A. $\frac{11}{9} \le R^{\infty}_{A} \le \frac{5}{4}$, and
B. $\frac{1}{m+2} - \frac{2}{m(m+1)(m+2)} \le R^{\infty}_{A}(\alpha) \le \frac{1}{m+2}$, where $m = \lfloor 1/\alpha \rfloor$.

The maximum discrepancy so far identified between Any Fit Decreasing algorithms is that between First Fit Decreasing and Worst Fit Decreasing, each of which can for certain classes of lists produce packings that are only 8/9 as long as the packings produced by the other [Joh73]. As might be expected, given the examples in Figure 2.3, the alternative of first sorting the items in *increasing* order by size is counterproductive, and indeed, any algorithm A that applies an Any Fit algorithm after first performing such a sort must have $R^{\infty}_{A} \ge T_{\infty} = 1.69103\ldots$

One potential drawback of First and Best Fit Decreasing is that they require $\Omega(n \log n)$ time to pack n items, both for the initial sorting phase and for the subsequent packing phase, at least if one assumes simple comparison-based implementations [Joh73]. How well can one do if one restricts attention to algorithms with linear-time implementations?

In the previous section, we have already seen one linear-time offline algorithm that beats the online bound (although there we were viewing it as a semi-online algorithm). This is the first algorithm of [GPT90], which had $R_A^\infty = 1.5$. The same worst-case bound was attained earlier by the more fully offline Group-X Fit Grouped algorithm (GXFG) of [Joh73] [Joh74]. This algorithm applies the Group-X Fit algorithm mentioned in Section 2.2.5 after first classifying the items by size using the set X of breakpoints and then reordering them so that items in each class go together, with the classes themselves being ordered by decreasing size. In [Joh73] [Joh74] it is shown that for all $m \geq 1$ if X contains $1/m$, $1/(m+1)$, and $1/(m+2)$ then for all $\alpha \leq 1$ such that $m = \lfloor 1/\alpha \rfloor$, $R_{\mathrm{G}X\mathrm{FG}}^\infty(\alpha) = 1+1/m$. For the special case of $\alpha = 1$, this yields $R_{\mathrm{G}X\mathrm{FG}}^\infty = 1.5$ when $X = \{1/2, 1/3, 1/4\}$.

If one reorders items according to this last set of breakpoints and then packs using an Any Fit packing rule, one gets $R_A^\infty = 4/3$ [Joh73], but this will not be a linear-time algorithm. One can, however, obtain an asymptotic worst-case ratio of 4/3 in linear time if one uses an algorithm due to Martel [Mar85]. This algorithm uses the breakpoint set $X = \{1/4, 1/3, /1/2, 2/3\}$ but does not easily fit into the two-part paradigm in which the list is first reordered according to a simple rule and then an online packing algorithm is applied. For details, see [Mar85].

No algorithm is currently known that does better than this 4/3 bound and runs in linear time on a sequential computer, although if one is fortunate enough to have an EREW PRAM with $n/\log n$ processors, one can in fact construct a packing in parallel $O(\log n)$ time that obeys the same worst-case bound as FFD (i.e, 11/9 asymptotically) [AMW89]. The algorithm performs a two-stage process, with the first stage constructing the FFD packing of all items with $s(a) > 1/6$ and the second stage adding the remaining small items to the packing efficiently. It is unlikely that the FFD packing for the full set of items can be precisely constructed in polylogarithmic parallel time, as constructing the FFD packing is a P-hard problem [AMW89].

Having seen what we can do with less than the $\Theta(n \log n)$ time required by FFD, the next question is what can we do if we are allowed more than $\Theta(n \log n)$ time? Can we devise algorithms with $R_A^\infty < 11/9$? This question was first answered in the positive by Yao [Yao80], who showed that there exists an $O(n^{10} \log n)$ time algorithm (called *Refined First Fit Decreasing* by Yao) with $R_A^\infty \leq 11/9 - 10^{-7}$.

This existence proof was later followed by more practical contenders. Garey and Johnson [GJ85] proposed an algorithm they called *Modified First Fit Decreasing* (MFFD) that improves on FFD much more substantially. It differs from First Fit Decreasing only in the way it packs the items with sizes in $(1/6, 1/3]$, but in handling these it also departs from the reordering-plus-online-packing paradigm. In packing these items, one considers the bins currently containing a single item of size exceeding 1/2 (the "A-bins" in the terminology of [GJ85]) from right to left, i.e., in order of decreasing gaps. To treat the current A-bin, one first checks if the two smallest still-unpacked items with size in $(1/6, 1/3]$ will fit together in the bin. If so, we place the smallest such item in the bin, together with the largest remaining such item that will fit with it. If not, this special phase is over, and all the remaining unpacked items are added to the current packing according to First Fit Decreasing. This algorithm has the same $O(n \log n)$ running time as FFD, and has roughly the same constant of proportionality. Its worst-case behavior is characterized as follows.

THEOREM 2.10 [GJ85]. $R^{\infty}_{\mathrm{MFFD}} = \frac{71}{60} = 1.18333....$

An alternative approach to beating FFD was subsequently proposed by Friesen and Langston [FL91]. They modify FFD as follows. First, note that the FFD packing can be constructed bin by bin using the following rule: To pack the next bin, continue adding the largest unpacked item that will fit until no such items remain. The algorithm Best Two Fit (B2F) also proceeds bin by bin, but now does some postprocessing after FFD is used to first fill the bin. If the bin contains more than one item at this point, we check to see if the smallest item in it can be replaced by the two smallest currently unpacked items with size 1/6 or greater. If so, it is replaced by the pair of such items that has the largest total size that will fit, and is added back to the list of unpacked items. This process continues until there are no unpacked items left of size greater than 1/6, at which point we revert to FFD. Friesen and Langston showed that $R^{\infty}_{\mathrm{B2F}} = 5/4 = 1.25$, which isn't better than what FFD can provide, but they also showed that the worst-case instances for the two algorithms were complementary. Thus, if we denote by CFB the compound algorithm that runs *both* B2F and FFD and outputs the better of the two packings, we do much better than either algorithm separately, in only twice the overall running time. More specifically, we have the following.

THEOREM 2.11 [FL91]. $1.164... = \frac{227}{195} \le R^{\infty}_{\mathrm{CFB}} \le \frac{6}{5} = 1.2$

MFFD and B2F try to improve the packings of individual bins by considering *pairs* of items as well as individual items in making their packing decisions. One might think that even better results would be possible (at the price of increased running time) if one was allowed to consider larger sets of items as units. With this idea in mind, Johnson [Joh73] proposed a sequence of algorithms *Most-k Fit* (MF_K), $k \ge 1$. As with B2F, these algorithms construct packings bin by bin. To pack the next bin, one starts by placing the largest as-yet-unpacked item in the bin. Then, so long as the smallest unpacked item will fit in the gap, one repeatedly adds the set of k or fewer items that will fit with the least space left over. Once the gap is too small for the smallest unpacked item, we go on to the next bin. The running time for MF_K is $O(n^k \log n)$ and hence not all that practical for k much greater than 2. Johnson observed that MF_2 did much better than FFD on its own worst-case examples, and conjectured that $\lim_{k\to\infty} R^{\infty}_{\mathrm{MF}_k}$ might equal 10/9. Unfortunately, the worst case examples of [FL91] for B2F imply that in fact $R^{\infty}_{\mathrm{MF}_K} \ge 5/4$ for all $k \ge 2$, so even going from $\mathrm{MF}_1 = \mathrm{FFD}$ to MF_2 causes a degradation in performance.

An even greater degradation occurs if one eliminates the requirement that the largest unpacked item start each bin. Graham [Gra72] has analyzed the algorithm that proceeds by packing each bin in turn so as to contain the set of unpacked items of largest total size no more than 1 (whether that set contains the largest item or not). Not only does this algorithm have to solve an NP-hard problem at each step, its worst-case behavior is much worse than that even for FFD. For this algorithm $R_A^\infty = \sum_{k=1}^{\infty} 1/(2^k - 1) \approx 1.60669...$[Gra72]. (The examples that imply this bound are described in [Joh73].)

As a final candidate for a simple algorithm that might have better worst-case behavior than MFFD, let us consider an algorithm proposed by Kenyon [Ken96]. This is a variant on Best Fit Decreasing that we might call *Best Fit Randomized* (BFR). Like BFD, this algorithm applies the Best Fit packing rule after first re-ordering the list, but instead of sorting the list, it simply performs a random permutation on it. Unlike our previous algorithms, Best Fit Randomized thus does not produce a unique packing for any given list, but a random distribution of packings. For such an algorithm A, a reasonable measure of its performance on a list of items L would be the *expected* number of bins it uses $E[A(L)]$. For worst-case behavior, one then looks for those lists L that maximize $E[A(L)]/\mathrm{OPT}(L)$, and define R_A^∞ correspondingly. Using these definitions, we have the following.

THEOREM 2.12 **[Ken96].** $1.08 = \frac{227}{195} \le R_{\mathrm{BFR}}^\infty \le 1.5$

It is likely that R_{BFR}^∞ is bigger than the stated lower bound. One can generate million-item lists L that, based on a sampling of permutations, appear to have $E[\mathrm{BFR}(L)]/\mathrm{OPT}(L) \approx 1.144$ [JKS95]. Nevertheless, 1.08 represents the largest ratio so far obtained for instances whose value of $E[\mathrm{BFR}(L)]$ can be analytically determined. No examples have been found that empirically yield expected ratios exceeding 1.144, however, so it is possible that in fact $R_{\mathrm{BFR}}^\infty < 1.15$.

This concludes our discussion of "practical" offline bin packing algorithms. If the only running time constraint one worries about is the theoretical one of polynomial time, one can do significantly better, as we shall see in the next section.

2.2.9 SPECIAL-CASE OPTIMALITY, APPROXIMATION SCHEMES, AND ASYMPTOTICALLY OPTIMAL ALGORITHMS

There are basically two approaches to improving on the worst-case ratios highlighted in the previous section. Before we discuss the approach that is the main subject of this section, let us briefly mention the other way to get better worst-case performance ratios. This is to restrict attention to specific types of input lists. For instance, for most of the algorithms we have considered so far, worst-case ratios approach 1 as the maximum item size approaches 0, i.e., $\lim_{\alpha \to 0} R_A^\infty(\alpha) = 1$.

A second class of instances that yield substantially improved worst-case results are those with *divisible item sizes*. As described in [CGJ87], a sequence of item sizes $s_1 > s_2 > \cdots > s_i > s_{i+1} > \cdots$ is a *divisible sequence* if for each $i > 1$, s_i exactly divides s_{i+1}. A list L of items is *weakly divisible* if the sizes of the items when sorted form a divisible sequence. It is *strongly divisible* if it is weakly divisible and the largest item size is of the form $1/k$ for some integer k, i.e., if it exactly divides the bin capacity. A natural example would be a list where all item sizes are of the form $1/2^j$ for various integers j. It is shown in [CGJ87] that, so long as L is weakly divisible, First Fit Decreasing always produces optimal packings, and if L is strongly divisible, then First Fit produces optimal packings as well. (Analogous results hold for many bin packing variants [CGJ87].)

Another class of instances that theoretically can be solved optimally in polynomial time are those in which the number of item sizes is bounded, independent of the number of items. If there are only k item sizes, and the smallest is bigger than $1/j$, then there are $O(k^{j-1})$ possible ways in which a bin can be filled by items of the various sizes, and hence at most $O(n^{k^{j-1}})$ possible packings, assuming we treat items of the same size as indistinct. This is a polynomial-bounded number of options, and one can in polynomial time check each one to see if it is feasible, given the number of items of each size in the given list. Obviously the actual list of potential packings can be substantially pruned, but even so, the resulting running time bound will still be far from a "low order polynomial."

From a theoretical point of view, there is a better way to solve this problem than simply trying all possible packings [BE83]. Determining the number of bins needed of each type can be formulated as an integer program (IP) with a variable for each bin type and constraints that insure that the total number of occurrences of items of size s is precisely the number of such items in the instance. This IP will have an A-matrix with k rows (constraints) and $O(k^{j-1})$ columns (variables). Because the number of variables is bounded, it can be solved in time polynomial in the number of constraints using the algorithm of [Len83] (although the time is exponential in the number of variables). Because the number of constraints is also bounded, the total time for constructing and solving the IP is thus just a constant, albeit one that is exponential in k and doubly exponential in j. To this constant must be added linear time for turning the solution of the IP into an explicit bin-by-bin description of the packing. Thus one can find an optimal packing in linear time. (We assume, as is standard in discussions of running times for bin packing algorithms, that the item sizes are rational numbers whose numerators and denominators are integers with binary representations that will fit in a single register of our computer, and that the registers are also big enough to contain the binary representation of the number n of items in the input.)

If one is willing to settle for asymptotic optimality rather than exact optimality, one can follow Gilmore and Gomory [GG61] [GG63] and simply solve the linear programming (LP) relaxation of the above IP. Using the ellipsoid method or a polynomial-time interior point algorithm along with appropriate post-processing techniques, one can find a basic optimal solution to the LP in time that is polynomial in k and only singly-exponential in j, a major reduction in the amount of "constant time" needed for the original IP. Such a basic solution will have at most k non-zero variables. These variables can then be rounded up to induce a packing of a superset of our original list, which can be converted in linear time to a bin-by-bin description of a packing of L that has at most k more bins than the optimal number.

By giving up a bit more in the worst-case guarantee, one can extend this approach to arbitrary instances, as shown by Fernandez de la Vega and Lueker.

THEOREM 2.13 [FL81]. For any $\epsilon > 0$, there exists a linear time algorithm A_ϵ such that $R^\infty_{A_\epsilon} \le 1+\epsilon$.

In standard terminology, the algorithms A_ϵ constitute an *approximation scheme* [GJ79] for one-dimensional bin packing, or more precisely an *asymptotic* approximation scheme, since here we deal with asymptotic as opposed to absolute worst-case ratios. Here is a sketch of how algorithm A_ϵ works, adapted from [FL81]. Note that to prove the theorem, we need only show that for all lists L, $A_\epsilon(L) \le (1+\epsilon)\cdot \mathrm{OPT}(L) + K_\epsilon$ bins for some constant $K_\epsilon \ge 1$ depending only on ϵ. (As we shall see, $K_\epsilon = 4/\epsilon$ will suffice.)

We may assume without loss of generality that $\epsilon \le 1$. For a suitably chosen $\epsilon' < \epsilon$, we begin by partitioning the given input list L into two parts L' and L_ϵ, where the latter part consists of all items a with $s(a) \le \epsilon'$ and can be identified in linear time. If we can pack L' into $(1+\epsilon)\cdot\mathrm{OPT}(L)+K_\epsilon$ bins, we will essentially be done, as the following procedure will suffice to construct an overall packing that satisfies the same bound: Add the elements of L_ϵ to the packing one bin at a time, continuing to add items to a bin until the next one does not fit. This is a linear-time operation. If no new bins are created, our final packing will still obey the desired bound. If a new bin *is* created, all bins except the last must be filled at least to the level $1-\epsilon'$, which implies that the total number of bins used is at most $\mathrm{OPT}(L)/(1-\epsilon')+1$. This will be less than $(1+\epsilon)\cdot\mathrm{OPT}(L)+1$ and hence less than $(1+\epsilon)\cdot\mathrm{OPT}(L)+K_\epsilon$ so long as $\epsilon' \le \epsilon/(1+\epsilon)$. (For future reference, note that because of this we may assume that $\epsilon' > \epsilon/2$.)

So let us concentrate on L'. Let $n' = |L'|$, $m = \lceil 4/\epsilon^2 \rceil$, and $h = \lfloor n'/m \rfloor$. Now pretend L' is sorted in nondecreasing order by size as $a_1, a_2, \ldots, a_{n'}$. Let $L_1 = (b_1, b_2, \ldots, b_{(m-1)h})$ be a list consisting of h items of size $s(a_{jh})$, $1 \le j \le m-1$, sorted in nondecreasing order. Note that we must have $s(b_i) \le s(a_{i+h})$, $1 \le i \le (m-1)h$, and so $\mathrm{OPT}(L_1) \le \mathrm{OPT}(L')$. Moreover, the items of L_1 are restricted to $m-1 < 4/\epsilon^2$ distinct sizes, all larger than $\epsilon/2$. Thus by the above argument, one can for fixed ϵ construct in linear time a packing for L_1 that uses at most $\mathrm{OPT}(L_1)+m-1 \le \mathrm{OPT}(L')+4/\epsilon$ bins. Moreover, note that for $1 \le i \le (m-1)h$, $s(b_i) \ge s(a_i)$, and so we can in linear time convert our packing of L_1 to a packing of the smallest $(m-1)h$ items in L' that uses just $\mathrm{OPT}(L')+4/\epsilon$ bins, leaving at most $2h-1$ items unpacked. If we pack these leftover items one per bin, we obtain an overall packing of L' that uses at most $\mathrm{OPT}(L')+2h-1+4/\epsilon$ bins. But by definition, $h \le (n')(\epsilon^2/4)$ and $OPT(L') \ge (n')(\epsilon') \ge (n')(\epsilon/2)$. Thus $2h-1 \le \epsilon\cdot\mathrm{OPT}(L')$, and we have constructed a packing of L' using at most $(1+\epsilon)\cdot\mathrm{OPT}(L')+4/\epsilon$ bins, as desired.

For fixed ϵ, the running time for this algorithm remains linear (assuming a model of computation as described above). This is because we do not actually have to sort L' in order to identify the key $m-1$ item sizes, which can be found by use of linear-time median-finding techniques. More precisely, the running time is $C_\epsilon + Cn\log(1/\epsilon)$, where C is a fixed constant independent of ϵ and C_ϵ is a constant reflecting the cost of constructing and solving the key linear program. Assuming we use the ellipsoid method or an appropriate polynomial-time interior point algorithm for solving the LP, C_ϵ should be polynomial in $(4/\epsilon)^{(1+\epsilon)/\epsilon}$. This is probably too large to yield feasible computations even for $\epsilon = .18333$, which would yield the same asymptotic worst-case ratio as MFFD.

Since we are being theoretical, however, it is interesting to note what happens when we let ϵ grow slowly with $\mathrm{OPT}(L) \leq n$. (We can determine close bounds on $\mathrm{OPT}(L)$ using FFD, so such growth can easily be arranged.) Suppose for instance that we let $\epsilon \approx (\log\log OPT(L))/(\log \mathrm{OPT}(L))$. The worst-case guarantee would then be

$$A(L) \leq (1+\frac{\log\log(\mathrm{OPT}(L))}{\log(\mathrm{OPT}(L))})\cdot \mathrm{OPT}(L)+\frac{4\log(\mathrm{OPT}(L))}{\log\log(\mathrm{OPT}(L))}$$

which would imply that $R_A^\infty = 1$. Moreover, for some fixed constant d, the running time would then be bounded by a polynomial in $d^{(\log\log n)(1+(\log n)/(\log\log n))}$, which is a (possibly high-order) polynomial in n. Thus we have the following corollary to the result of [FL91], first observed in [Joh82].

THEOREM 2.14. Polynomial-time bin packing algorithms exist with $R_A^\infty = 1$.

A much better guarantee of this sort was subsequently discovered by Karmarkar and Karp [KK82], who demonstrated a polynomial-time algorithm A that guarantees $A(L) \leq \mathrm{OPT}(L) + \log^2(\mathrm{OPT}(L))$. They exploit many additional techniques beyond those in [FL81] to obtain this result. One key idea is to avoid actually generating the entire linear program. For this they concentrate on the dual of the LP described above (actually, the dual of an LP that is similar to that of [FL81] but is constructed using somewhat more sophisticated rounding procedures). In the dual, the variables correspond to items and the constraints correspond to the packings. Karmarkar and Karp then apply the ellipsoid method, which only generates constraints as they are needed for separation purposes. Moreover, tbese separating constraints turn out to be solutions to knapsack problems, as in the column-generation approach that Gilmore and Gomory proposed decades ago for solving the original problem using the simplex method [GG61] [GG63]. The knapsack problems are NP-hard, but fortunately we can settle for near-optimal as opposed to optimal solutions in this context, since we only need to solve the LP to within an additive constant. Thus existing polynomial-time approximation algorithms for the knapsack problem can be used. Once the LP has been approximately solved, Karmarkar and Karp exploit additional ideas to eliminate unneeded constraints (bin-types) so as to get down to a near-basic solution, from which a final packing can be derived.

Unfortunately, all this cleverness comes at a price, and the best running-time bound proved in [KK82] is worse than $O(n^8)$. Better running times are possible, however, if one is willing to settle for less than asymptotic optimality. Whereas in the [FL81] scheme a guarantee of the form $A_\epsilon(L) \leq (1+\epsilon)\cdot \mathrm{OPT}(L) + K_\epsilon$ required time exponential in $1/\epsilon$, an analogous guarantee based on the [KK82] approach can be obtained in running time bounded by a polynomial in $1/\epsilon$ (although the dependence on n is $\Theta(n \log n)$ rather than linear, as it was in the [FL81] scheme). Thus we have a fully polynomial (asymptotic) approximation scheme (FPAS) for bin packing. For full details see [KK82]. An alternative sketch of the results of [FL81] and [KK82] is given in Chapter 9.

From the theoretical point of view, the Karmarkar-Karp algorithm leaves little room for further improvement in the level of approximation obtainable by polynomial-time algorithms. We should point out, however, that there are to date no NP-hardness results that rule out the possibility (assuming P $\neq$ NP) of a polynomial-time heuristic A that guarantees $A(L) \leq \mathrm{OPT}(L) + d$ for some fixed constant d.

2.2.10 OTHER WORST-CASE QUESTIONS

2.2.10.1 Absolute worst-case ratios

The results we have mentioned so far concentrate on *asymptotic* worst-case ratios R_A^∞, but the question of absolute worst-case ratios $R_A \equiv \inf\{r \geq 1 : R_A(L) \leq r$ for all lists $L\}$ is also of interest, especially when we are considering relatively short lists of items. For these the current best results are as follows:

THEOREM 2.15 [Sim94].

A. For $A \in \{\mathrm{FF}, \mathrm{BF}\}$, $R_A \leq 1.75$.

B. For $A \in \{\mathrm{FFD}, \mathrm{BFD}\}$, $R_A = 1.5$.

Note that the former bound is actually quite close to the asymptotic bound of $R_{\mathrm{FF}}^\infty = R_{\mathrm{BF}}^\infty = 1.7$. The latter is further from the asymptotic bound of $R_{\mathrm{FFD}}^\infty = R_{\mathrm{BFD}}^\infty = 11/9$, but in comparing the latter conclusion to the specific results underlying the asymptotic conclusion, we see that Theorem 2.15 provides a better bound at least for $\mathrm{OPT}(L) \leq 3$. To be specific, if it is true that $\mathrm{FFD}(L) \leq (11/9) \cdot \mathrm{OPT}(L) + \alpha$ for all L, then the absolute bound is stronger only for $OPT(L) \leq (18/5)\alpha$. Baker shows in [Bak83] that $\alpha = 3$ suffices, implying that the asymptotic result dominates for $OPT(L) > 10$. In [Yue91] a proof is claimed for $\alpha = 1$, which would imply that the asymptotic result dominates the absolute one for $\mathrm{OPT}(L) > 3$. More recently, [CSB94] presents a tighter result that dominates both [Yue91] and [Sim94] when $\mathrm{OPT}(L) \in \{3, 4, 5\}$. Using techniques that exploit linear programming lower bounds on the optimal packing, the authors prove that both $\mathrm{FFD}(L)$ and $\mathrm{BFD}(L)$ are no more than $(4/3) \cdot \mathrm{OPT}(L) + 1/3$.

2.2.10.2 Bounds on anomalous behavior

One difficulty that often must be confronted in proving worst-case guarantees for bin packing algorithms is the fact that certain natural monotonicity properties can be violated by the better heuristics. Let us say that a list L_2 is *dominated* by a list L_1 if L_2 can be obtained from L_1 by deleting items and/or reducing their size. A bin packing algorithm A is *monotone* if for any two lists L_1 and L_2 where L_1 dominates L_2, $A(L_2) \leq A(L_1)$ [Mur88]. Such a property clearly might be helpful in proving results about algorithm A. Indeed, in Section 2.3 we shall introduce a monotone algorithm *Matching Best Fit* as a key intermediary in proving results about the average-case behavior of the non-monotone Best

Fit algorithm. Unfortunately few of the algorithms we have discussed so far are monotone. In a comprehensive study, Murgolo [Mur88] was able to identify only two: Next Fit and Next-2 Fit. Monotonicity does not hold even for Next-3 Fit or for ABF_2 (the algorithm that replaces the First Fit packing rule in Next-2 Fit by the Best Fit rule). Nor does it hold for such simple algorithms as the Harmonic algorithms H_K, $K \geq 3$ (although it holds trivially for H_1 and H_2).

Given that most of the algorithms worth studying are non-monotonic, the question then arises, just how non-monotonic can they be? For the four most famous bin packing heuristics, FF, BF, FFD, and BFD, the non-monotonicity can be arbitrarily large, that is it can grow with n. This was implicit in examples discovered by Halász [Hal74], but has been studied in most detail by Murgolo, who constructed lists implying the following result. If A is a bin packing algorithm, define the *asymptotic worst-case nonmonotonicity* of A, denoted by N_A^∞ to be the maximum value ϕ such for all $N > 0$ there exists a pair of lists (L_1, L_2) with $\mathrm{OPT}(L_1) \geq N$, L_1 dominating L_2, and $A(L_2)/A(L_1) - 1 \geq \phi$. In other words, the nonmonotonicity is the asymptotic fraction by which the number of bins may increase when items are shrunk or deleted.

THEOREM 2.16 [Mur85][Mur88]. With respect to asymptotic worst-case nonmonotonicity,

A. Both N_{WF}^∞ and N_{WFD}^∞ are at least $1/15$,

B. Both N_{FF}^∞ and N_{BF}^∞ are at least $1/42$, and

C. Both N_{FFD}^∞ and N_{BFD}^∞ are at least $1/75$.

In general, no upper bounds on nonmonotonicity besides the trivial that $N_A^\infty \leq R_A^\infty - 1$ are currently known. For more details on nonmonotonicity, see [Mur85] [Mur88].

2.3 AVERAGE-CASE ANALYSIS

One drawback of relying on worst-case analysis is that in many applications the worst case never seems to occur. Indeed, quite intricate and unlikely lists of items were needed to prove many of the lower bounds in the previous section. Thus there is a strong need for results that tell us more about typical behavior. Proving results about the average-case behavior of heuristics under a variety of item-size distributions is one step in this direction. Objections can be lodged against any single choice of distribution, since real-world applications rarely generate instances that obey the sorts of probability distributions for which tight analysis is currently possible. Objections might also be made to the standard assumption that each item size is chosen independently of all the rest. Nevertheless, if results are obtained for a variety of such distributions, a broader picture may begin to emerge, and at the very least the worst-case results can be put into perspective. In the case of the classical one-dimensional bin packing problem, just such a broad-based average-case approach has begun to emerge, and we survey it in this section. In spite of the difficulty of probabilistic analysis, both the general theory and average-case results for specific algorithms have grown impressively in the past decade.

Let us begin with some terminology. Although a variety of item-size distributions have been studied, the common assumption is that the items in a list have independent, identically distributed sizes. Let F denote the item-size distribution, and denote its mean and variance by μ and σ^2. For purposes of asymptotic analysis, we are typically concerned with a list L_n of n randomly generated items. For simplicity, our notation does not specify what distribution F is under consideration, but the identity of F will always be made clear.

In this context, many of our standard functions of lists become random variables: $s(L_n)$ (the sum of the item sizes), $\mathrm{OPT}(L_n)$, $A(L_n)$ (where A is a bin packing algorithm), and $R_A(L_n) \equiv A(L_n)/\mathrm{OPT}(L_n)$. We are also interested in the derived random variable $W_A(L_n) \equiv A(L_n) - s(L_n)$, which is the amount of wasted space in the bins of the packing of L_n by algorithm A. It is often easier to prove average-case results about $W_A(L_n)$ than about $R_A(L_n)$ directly. Fortunately, the former typically imply the latter, since for most of the distributions that have been studied $\lim_{n\to\infty} E[\mathrm{OPT}(L_n)/s(L_n)] = 1$. Moreover, even for those distributions F and algorithms A where $\lim_{n\to\infty} E[A(L_n)/s(L_n)]$ also equals 1, we typically have $E[W_{\mathrm{OPT}}(L_n)] = o(E[W_A(L_n)])$, so that $E[W_A(L_n)] = \Theta(E[A(L_n) - \mathrm{OPT}(L_n)])$.

In what follows, we use average-case notation that mirrors the worst-case notation of Section 2.2. For a given item-size distribution F and algorithm A, define

$$\bar{R}_A^n(F) \equiv E[R_A(L_n)] = E\left[\frac{A(L_n)}{\mathrm{OPT}(L_n)}\right]$$

and

$$\bar{W}_A^n(F) \equiv E[A(L_n) - s(L_n)]$$

for lists L_n generated according to F. We then can define the *asymptotic expected ratio* for A under F to be

$$\bar{R}_A^\infty(F) \equiv \lim_{n\to\infty} \bar{R}_A^n(F)$$

(No asymptotic notation is needed for expected wasted space, as this is unbounded for most algorithms and distributions.)

Although there have been some general average-case results that hold for arbitrary distributions F, most of the specific expected values that have been computed concern *continuous uniform* or *discrete uniform* distributions. A distribution of the former type is denoted by $U[a,b]$, where $0 \le a < b \le 1$, and item-sizes are chosen uniformly from the (continuous) interval $[a,b]$. Typically $a = 0$, although we shall present some results for $a > 0$ in Section 2.3.5. Distributions of the second type are denoted by $U\{j,k\}$, $1 \le j \le k$, where item sizes are chosen uniformly from the finite set $\{1/k, 2/k, \ldots, j/k\}$. Note that as m approaches ∞, the discrete distributions $U\{jm, km\}$ become more and more like the continuous distribution $U[0, j/k]$. As we shall, see, however, this does not prevent the two types of distribution from yielding results that are mathematically very different.

Our discussion of the average case parallels our earlier discussion of worst-case results, and we organize the presentation by algorithm rather than by distribution. In Section 2.3.1 we survey average-case results for bounded-space online algorithms. Section 2.3.2 covers results for arbitrary online algorithms, with special emphasis on the much-studied First and Best Fit algorithms. Section 2.3.3 describes results for the offline case. These three sections all concentrate on questions about $\bar{R}_A^\infty(F)$ and $\bar{W}_A^n(F)$. Section 2.3.4 considers two other average-case questions that have received attention: (1) what is the expected optimal number of bins under various distributions and (2) what can we say about the probability distribution of $A(L_n)$ as a function of n for various algorithms and distributions. The results relating to question (1) help back up the claims made above about the relationship between $E[s(L_n)]$ and $E[\mathrm{OPT}(L_n)]$. The results relating to question (2) allow us to reach conclusions about $\bar{R}_A^\infty(F)$, which measures expected ratios, from results about ratios of expectations, something we shall do regularly in Sections 2.3.1 through 2.3.3, and only justify when we get to Section 2.3.4.2. We shall not provide much detail about proof techniques beyond highlighting the different types of approaches taken. Readers interested in learning more details about these approaches are referred to the corresponding references and to the monograph by Coffman and Lueker [CL91].

2.3.1 BOUNDED-SPACE ONLINE ALGORITHMS

The first probabilistic analysis of bin packing algorithms appears to be that of Shapiro [Sha77], who proposed an approximate analysis of Next Fit based on the exponential distribution. The first precise average-case asymptotics were those of Coffman, So, Hofri, and Yao [CSH80], again for Next Fit. They proved that, if $F = U[0,1]$, then the distribution of the level of the highest indexed closed bin converges geometrically fast to the stationary distribution $V(x) = x^3, 0 \le x \le 1$ with mean 3/4. From this we can conclude the following.

THEOREM 2.17 [CSH80]. $\bar{R}_{\mathrm{NF}}^\infty(U[0,1]) = \frac{4}{3}$.

Karmarkar [Kar82a] subsequently extended the NF results to cover the distributions $U[0,b]$. His analysis came down to the solution of linear systems of differential equations. Closed-form results were obtained for $1/2 \le a \le 1$ and showed that $E[\mathrm{NF}(L_n)] \sim \rho n$ as $n \to \infty$, where

$$\rho = \frac{1}{12b^3}(15b^3 - 9b^2 + 3b - 1) + \sqrt{2}\left(\frac{1-b}{2b}\right)^2 \tanh\left(\frac{1-b}{\sqrt{2}b}\right),$$

which gave numbers confirming the experimental results of Ong, Magazine, and Wee [OMW84]. A numerical calculation using this expression verified that, asymptotically, the packing efficiency $\bar{R}_{\mathrm{NF}}^\infty(U[0,b])$ is not monotone in b as one

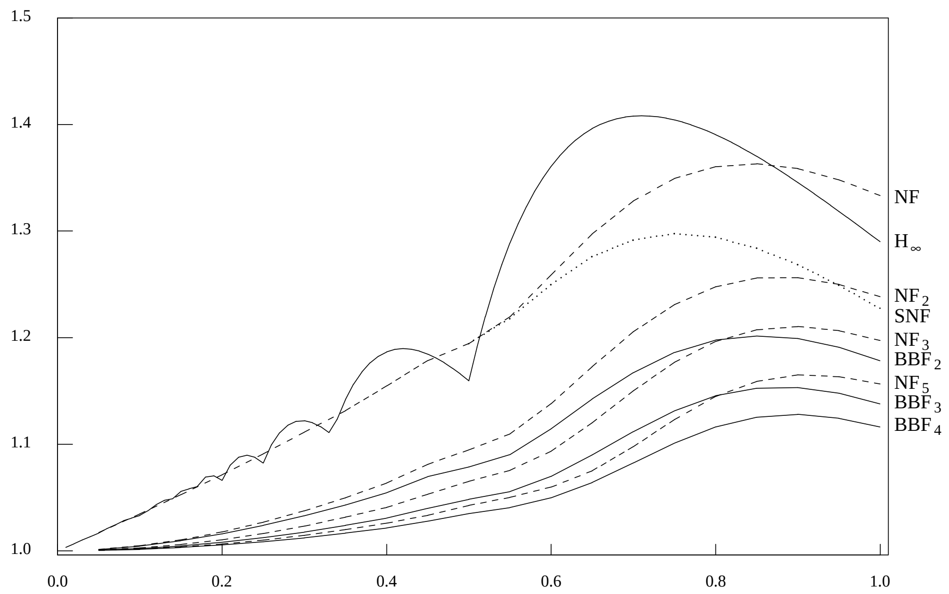

FIGURE 2.6

$\bar{R}_A^\infty(U[0,b])$ as a function of b.

might expect; it reaches a maximum at $b \approx 0.841$.

The average-case performance of the Harmonic algorithms H_K is relatively easy to obtain for general distributions F. Lee and Lee [LL87] showed that, for all $K \geq 3$, $\bar{R}_{\mathrm{H}_K}^\infty(U[0,1]) < 1.306$, and hence is significantly better than the asymptotic expected ratio of $4/3$ for Next Fit, even for relatively small K. As K goes to ∞ we have

THEOREM 2.18 [LL87][CL91]. $\lim_{K\to\infty} \bar{R}_{\mathrm{H}_K}^\infty(U[0,1]) = \pi^2/3 - 2 \approx 1.2899.$

Note however that even the limiting difference of $4/3 - 1.2899$ is far smaller than the 2.0 versus 1.691... worst-case gap between the two algorithms. For a pictorial comparison of the results for Next Fit and the Harmonic algorithms, see Figure 2.6, which originally appeared in [CJ91] and covers results for $F = U[0,b]$, $0 < b \leq 1$. The curve labeled H_∞ in the figure represents $\lim_{K\to\infty} \bar{R}_{\mathrm{H}_K}^\infty(U[0,b])$. Note that the expected values for H_∞ are not uniformly better than those for Next Fit. Indeed, they oscillate as a function of b. The local minima occur whenever b is of the form $1/m$ for m an integer. Values for Next Fit when $b < 1/2$ are experimentally determined, as are the curves included for some of the other algorithms of Section 2.2.6.

The figure also includes the curve for an interesting modification of Next Fit devised and analyzed by Ramanan [Ram89], called *Smart Next Fit* (SNF). This algorithm works as follows: The next item p to be packed is put by SNF into the current bin B if it fits, as in NF. But if p does not fit in B, then SNF puts p into a new bin B' and retains as the new current bin whichever of B and B' has the most space remaining. (Note that this algorithm lies somewhere between Next Fit and the 2-Bounded Best Fit algorithm of Section 2.2.6.) Ramanan applied Karmarkar's techniques and showed that improvements are possible under SNF when $F = U[0,b]$ with $1/2 < b \leq 1$. For $b = 1$, $\bar{R}^{\infty}_{\mathrm{SNF}}(U[0,b])$ is roughly 1.227 versus 1.333 for Next Fit, although the difference declines to 0 as $b \rightarrow 1/2$.

The curves in the figure for versions of Next-K Fit (NF_K) and K-Bounded Best Fit (BBF_K) raise interesting issues. Presumably as K goes to ∞ these curves should converge to those for First and Best Fit respectively, and one wonders just what those limiting curves will look like, given the rapid improvement (and flattening) of the curves depicted here. We will return to this question in the next section.

In the meantime we should note that, just as there are lower bounds on how well bounded-space online algorithms can perform in the worst-case, there are also nontrivial bounds on what they can do on average. In particular, no bounded-space online algorithm A can have $\bar{R}^{\infty}_{A}(U[0,1]) = 1$. Coffman and Shor [CS93] proved that such algorithms inevitably waste a constant fraction of the allocated space on average. More specifically, they show the following.

THEOREM 2.19 [CS93]. If A is an online algorithm limited to K active bins, then

$$\bar{W}^{n}_{A}(U[0,1]) \geq \frac{n}{16(K+1)}$$

Analogous results (with different constants of proportionality) also hold for the distributions $U[0,b]$, $0 < b < 1$. The lower bound of Theorem 2.19 can probably be improved substantially by tighter analysis, and it remains to be seen whether a tight lower bound can be obtained.

As to other sorts of distributions, results for discrete uniform distributions $U\{j,k\}$ should roughly track those for the corresponding continuous uniform distributions. There has been work on the average-case behavior of Next Fit under further sorts of distributions, for example the abovementioned study by Shapiro [Sha77] along with that of Halfin [Hal89], which were based on the exponential distribution, but these dealt with different questions and did not yield results that are directly comparable to the ones reported here.

2.3.2 ARBITRARY ONLINE ALGORITHMS

2.3.2.1 Results for $U[0,1]$

As pointed out in Section 2.2.5, relaxing the bounded-space requirement allows one to design algorithms with substantially better worst-case behavior than any bounded-space online algorithm. One of the improved algorithms covered in that section, the Modified Harmonic algorithm MH_{38} of Ramanan, Brown, Lee, and Lee [RBL89], has been analyzed from an average-case point of view.

In [RT89f], Ramanan and Tsuga show that this algorithm improves on the basic Harmonic algorithms on average as well as in the worst case, at least when $F = U[0,1]$. In particular, they show that $\bar{R}^{\infty}_{\mathrm{MH}_{38}}(U[0,1])$ lies between 1.27684 and 1.27688. This is significantly less than the 1.333... ratio for Next Fit and the 1.2899... limiting value for the Harmonic algorithms, although not as good as the ratio of 1.227 for Smart Next Fit. However, Ramanan and Tsuga then describe a sequence of variants on the Modified Harmonic algorithms (obtained simply by altering certain parameters but not the boundaries of the item-size classes) which have $\bar{R}^{\infty}_{\mathrm{MH}_K}(U[0,1]) < 1.2$ as soon as $K \geq 5$, and $\lim_{K\to\infty} \bar{R}^{\infty}_{\mathrm{MH}_K}(U[0,1]) \approx 1.189$.

This last-mentioned sequence of algorithms obtains its improved average-case performance at the cost of degraded worst-case performance with respect to MH_{38}. At a further cost in worst-case behavior, one can improve average-case performance still more. Indeed, as first observed by Hoffman [Hof82], there exist $\Theta(n \log n)$ time online algorithms whose expected behavior when $F = U[0,1]$ is in a sense asymptotically optimal. We illustrate Hoffman's approach using a variant studied by Coffman and Lueker [CL91, pp. 148–150]. The approach actually involves yet another indexed sequence of algorithms, which we shall call the *Online Match* algorithms (OM_K for $K \geq 3$) and which are closely related to the Interval First Fit algorithms of Csirik and Galambos [CG86]. Call p and p' *companions* for algorithm OM_K if $1 - 1/K \leq s(p) + s(p') \leq 1$. Algorithm OM_K maintains two initially empty packings; one is *partial match* packing, and the other is a NF packing that contains only items $\leq 1/2$. The algorithm packs the next item p according to the following two-part rule: If $s(p) \geq 1/2$, pack p into an empty bin and add the bin to the partial match packing. If $s(p) < 1/2$ and there exists a bin B in the partial match packing that contains only a companion of p, pack p in B along with its companion. If $s(p) < 1/2$ and no such bin exists, pack p into the NF packing. It is easily verified that the asymptotic worst-case ratio $R^{\infty}_{\mathrm{OM}_K} = 2$. The average case result is as follows.

THEOREM 2.20 [CL91]. For some universal constant c,

$$\bar{R}^{\infty}_{\mathrm{OM}_K}(U[0,1]) = 1 + c\sqrt{\frac{K}{n}} + O\left(\frac{1}{K}\right)$$

Thus $\lim_{K\to\infty} \bar{R}^{\infty}_{\mathrm{OM}_K}(U[0,1]) = 1$, although of course for each particular value of K we still have $\bar{R}^{\infty}_{\mathrm{OM}_K}(U[0,1]) > 1$. Note that we could presumably get a single algorithm with $\bar{R}^{\infty}_{A}(U[0,1]) = 1$ by letting K grow with N, say by setting $K = \log n$. Strictly speaking, however, this would be a violation of the online restriction, since it would require us to know n in advance. Let us call an algorithm that is online except for the fact that it knows n in advance a *closed* online algorithm. Thus it appears that there exist closed online algorithms A with $\bar{R}^{\infty}_{A}(U[0,1]) = 1$.

The question thus becomes whether there exist *open* online algorithms, i.e., ones that do not know n in advance, with this same optimality property. The answer is yes. One could devise such an algorithm from the Online Match approach above by adaptively increasing K as more items arrive, but fortunately something much simpler will suffice. In 1984, Bentley, Johnson, Leighton, McGeoch, and McGeoch [BJL84] proved the following surprising result.

THEOREM 2.21 [BJL84]. $\bar{R}^{\infty}_{\mathrm{FF}}(U[0,1]) = 1.$

This result was viewed as surprising because of earlier conjectures based on simulation results. Johnson in [Joh73] had conjectured based on samples lists with $n = 200$ that $\bar{R}^{\infty}_{\mathrm{FF}}(U[0,1]) \approx 1.07$. Ong, Magazine, and Wee [OMW84], based on sample lists with n ranging from 40 to 1000, updated this conjecture to $\bar{R}^{\infty}_{\mathrm{FF}}(U[0,1]) \approx 1.056$, still quite far from 1. It turned out, however, that much larger values of n were needed to get a true picture of asymptotic behavior. For instance, based on much more extensive experiments [BJL83] we now know that $\bar{R}^{n}_{\mathrm{FF}}(U[0,1])$ still exceeds 1.01 for n as large as 250,000. The question thus becomes how fast does $\bar{R}^{n}_{\mathrm{FF}}(U[0,1])$ approach 1.

This question can be addressed by turning our attention to expected waste, in particular the function $\bar{W}^{n}_{\mathrm{FF}}(U[0,1])$. It was shown by a complicated argument in [BJL84] that this function was $O(n^{0.8})$. Much simpler arguments have since been derived and we now know the precise order of the function.

THEOREM 2.22 [Sho86][CJS95]. $\bar{W}^{n}_{\mathrm{FF}}(U[0,1]) = \Theta(n^{2/3}).$

The algorithm Best Fit turns out to be even better:

THEOREM 2.23 [Sho86][LS89]. $\bar{W}^{n}_{\mathrm{BF}}(U[0,1]) = \Theta(\sqrt{n}\log^{3/4} n).$

The proofs of these results embody fascinating connections to the theory of stochastic planar matching, a theory that has had a surprising variety of applications. The foundation for the modern theory of stochastic planar matching is the work of Ajtai, Komlós, and Tusnády [AKT84]. Chapter 3 in [CL91] gives a broad coverage of the area, but the definitive treatment can be found in the more recent work of Talagrand [Tal94].

There are many variants on the original planar matching problem of [AKT84], several of which come up in bin packing contexts. As a primary example, let us consider the *up-right matching* problem. An instance of this problem is a set of n points chosen independently and uniformly at random in the unit square. The points are colored red and blue, with each point having equal probability of receiving either color, and the choices for all points being independent. We ask for a maximum matching of blue points to red points such that in each matched pair the blue point is above and to the right of the red point, i.e., in both coordinates, the blue point is at least as far from the origin as the red point. This problem originated in an analysis of a two-dimensional bin packing problem by Karp, Luby, and Marchetti-Spaccamela [KLM84b], who were the first to apply stochastic planar matching problems to bin packing.

The connection is to Best Fit and comes about as follows. Consider a random list L_n generated according to $F = U[0,1]$. An item a_i is identified with a blue point at location $(s(a_i), i/n)$ if $s(a_i) \le 1/2$ and with a red point at location $(1 - s(a_i), i/n)$ if $s(a_i) > 1/2$. An edge in an up-right matching thus corresponds to a pair of items such that the first-arriving has size exceeding 1/2 and the total size of the two items does not exceed 1, i.e., the pair can fit in a bin together. Thus the following simplified variant on Best Fit, called *Matching Best Fit* (MBF), will construct an up-right matching: Proceed as in Best Fit, except that bins are closed as soon as they receive an item a with $s(a) \le 1/2$. It is easy to see that the number of unmatched points that result will be roughly proportional to the amount of wasted space in the MBF packing of L_n. Shor proves three additional key facts about Matching Best Fit in [Sho86]:

- MBF constructs an *optimal* up-right matching.
- $\mathrm{MBF}(L) \ge \mathrm{BF}(L)$ for all lists L.
- $\bar{W}^n_{\mathrm{BF}}(U[0,1]) = \Omega(\bar{W}^n_{\mathrm{MBF}}(U[0,1]))$ so long as the latter is $\Omega(\sqrt{n \log n})$.

Thus, assuming that $\bar{W}^n_{\mathrm{MBF}}(U[0,1])$ grows quickly enough, we will have $\bar{W}^n_{\mathrm{BF}}(U[0,1]) = \Theta(\bar{W}^n_{\mathrm{MBF}}(U[0,1])) = \Theta(E[U_n])$, where U_n is defined to be the number of unmatched points in an optimal up-right matching on $2n$ random points. The initial results in [KLM84b] gave an $\Omega(\sqrt{n \log n})$ lower bound and a $O(\sqrt{n} \log n)$ upper bound on $E[U_n]$. A lower bound of $\Omega(\sqrt{n} \log^{3/4} n)$ was proved by Shor in [Sho86] and the tight upper bound $E[U_n] = O(\sqrt{n} \log^{3/4} n)$ was proved by Leighton and Shor in [LS89], thus completing the proof of Theorem 2.23. Rhee and Talagrand [RT88a] independently proved the same bounds on $E[U_n]$. Simpler proofs were discovered later by Coffman and Shor [CS91] and then, in a more general setting, by Talagrand [Tal94].

The proof of Theorem 2.22 for First Fit is derived in a similar fashion, based on an appropriately defined variant on up-right matching and using an analogously defined *Matching First Fit* algorithm as an intermediary. The $\Omega(n^{2/3})$ lower bound for $\bar{W}^n_{\mathrm{FF}}(U[0,1]$ was proved by Shor in [Sho86]; Shor's original $O(n^{2/3} \log n)$ upper bound was subsequently tightened to $O(n^{2/3})$ by Coffman, Johnson, Shor, and Weber in [CJS95].

Stochastic planar matching results can also be used to obtain lower bounds on the best possible average case performance for an online algorithm when $F = U[0,1]$. Here the relevant variant is the *rightward matching* problem. Instances are as in up-right matching, but now a point P can be matched either to a point of the other color, in which case the blue point must be to the right of the red point, or to a point directly beneath P on the lower boundary of the square. The objective is a matching with a minimum total vertical distance between matched points, i.e., a matching such that the sum of the vertical components of straight-line segments connecting matched points is minimized. Let us call this quantity the *vertical discrepancy* of the matching. Shor showed in [Sho86] that any open online algorithm operating a list L_n generated according to $F = U[0,1]$ will generate a rightward matching whose vertical discrepancy roughly equals the average wasted space during the course of the packing. Thus if we let V_n denote the vertical discrepancy of an optimal rightward matching for a random point set, we have $\bar{W}^n_A(U[0,1]) = \Omega(E[V_n])$ for all open online algorithms A. In [Sho86] he proved that $E[V_n] = \Omega(\sqrt{n \log n})$. In [Sho91] he exhibited an $O(n \log n)$-time open online algorithm that achieved this bound on wasted space. Thus we have the following.

THEOREM 2.24 [Sho86][Sho91]. For any open online algorithm A,

$$\bar{W}_A^n(U[0,1]) = \Omega(\sqrt{n \log n})$$

and this is the best possible such lower bound.

The above bound does not hold for closed online algorithms (ones that know n in advance). For these the only lower bound is the trivial one of $\bar{W}_A^n(U[0,1]) = \Omega(\sqrt{n})$ (trivial because $\bar{W}_{\mathrm{OPT}}^n(U[0,1]) = \Omega(\sqrt{n})$, as is easily seen). Moreover, as is shown in [Sho86], this bound is achieved by the simple algorithm that begins by packing the first $\lfloor n/2 \rfloor$ items one per bin, and then packs the remaining items using Best Fit. Note however, that there are drawbacks to algorithms that optimize their behavior for a particular distribution like $U[0,1]$. Both this algorithm and the one in Theorem 2.24 have unbounded worst-case behavior and indeed can have arbitrarily bad average-case behavior under natural distributions. For example, each has $\lim_{b \to 0} \bar{R}_A^\infty(U[0,b]) = \infty$.

2.3.2.2 Results for $U[0,b]$, $0 < b < 1$

Although the distribution $F = U[0,1]$ gives rise to interesting behavior and appealing mathematical connections, the very fact that simple matching algorithms suffice to yield good average-case behavior makes this distribution somewhat suspect. In real-world applications, one often has to put more than two items in a bin to get good packings. As a first step to obtaining a more realistic picture of real-world behavior, let us see what has been learned about average-case behavior under the distributions $F = [0,b]$, $0 < b < 1$.

For such distributions $\bar{W}_{\mathrm{OPT}}^n(F) = O(1)$, as we shall see in the next section, but a lower bound of $\Omega(n^{1/2})$ on $\bar{W}_A^n(F)$ holds for all open online algorithms A [CCG91]. No algorithm achieving this bound has yet been found, although one can come close. Rhee and Talagrand [RT93b, RT93c] have shown that, with only a slight weakening, results similar to those for Best Fit can be derived for any distribution F. They exhibit an online algorithm in [RT93c] that packs L_n into at most $\mathrm{OPT}(L_n) + K\sqrt{n}\log^{3/4} n$ bins with a probability at least $1 - n^{-\alpha\sqrt{\log n}}$, where K and α are universal constants, and where, as one might suspect, the $\sqrt{n}\log^{3/4} n$ term comes from the algorithm's connection with Best Fit. The algorithm works for all F, dynamically changing its packing strategy as it 'learns' more about F from items already packed. For the case of $F = U[0,b]$, $b < 1$, the above bound implies that $\bar{W}_A^n(F) = \Theta(\sqrt{n}\log^{3/4} n)$. Unfortunately, even if one is able to specialize the algorithm in advance to the distribution in question, it is still likely to be complex, it will only come into its own for huge values of n, and it will behave poorly in a worst-case sense.

More natural and robust alternatives have yet to be found. First and Best Fit in particular do not appear to be good candidates. Based on extensive simulations [BJL83] [Joh96], it appears that for all b, $0 < b < 1$ both Best and First Fit have linear expected waste, i.e., $\bar{W}_A^n(U[0,b]) = \Theta(n)$ and hence $\bar{R}_A^\infty(U[0,b]) > 1$. See Figure 2.7, which illustrates the case for First Fit and $.70 \leq b \leq 1$. The figure presents experimentally-derived curves for $\bar{R}_{\mathrm{FF}}^n(U[0,b])$ with b increasing

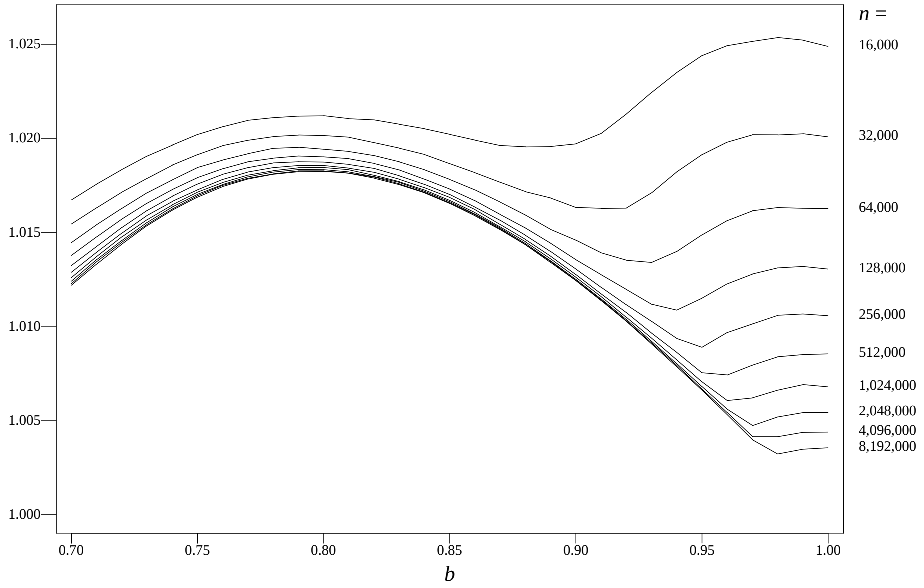

FIGURE 2.7

$\bar{R}^n_{\mathrm{FF}}(U[0,b])$ as a function of b for increasing values of n.

in increments of .01 from .70 to 1.00, and with n increasing by factors of 2 from 16,000 to 8,192,000. Note that the curves are consistent with the above conclusion about $\bar{R}^n_{\mathrm{FF}}(U[0,b])$, but also suggest (as one would hope given Theorem 2.21) that $\lim_{b\to 1} \bar{R}^\infty_{\mathrm{FF}}(U[0,b]) = 1$. As was the case for Next Fit, the value of $\bar{R}^\infty_{\mathrm{FF}}(U[0,b])$ takes on its maximum for b roughly equal to .81, although now the maximum value is quite small, i.e, less than 1.02. In general the values are so small that for small n they are overwhelmed by the $\Theta(n^{2/3})$ expected waste term, with $\bar{R}^n_{\mathrm{FF}}(U[0,b])$ actually increasing as $b \to 1$.

The cases for Best Fit and for $0 < b < .70$ are similar, except that $\bar{R}^\infty_{\mathrm{BF}}(U[0,b])$ appears always to be a bit less than $\bar{R}^\infty_{\mathrm{FF}}(U[0,b])$ for any fixed $b < 1$, and both approach 1 in the limit as b approaches 0. Unfortunately, although the evidence of the experiments reported in Figure 2.7 seems incontrovertible, no one has so far been able to rigorously prove that either $\bar{R}^\infty_{\mathrm{FF}}(U[0,b])$ or $\bar{R}^\infty_{\mathrm{BF}}(U[0,b])$ exceeds 1 for any b, $0 < b < 1$. To find proofs of such behavior we must turn to the analogous discrete uniform distributions $U\{j,k\}$ defined at the beginning of Section 2.3. As was the case for the bounded-space online algorithms of Section 2.3.1, the behavior of First and Best Fit on such distributions roughly mimics that for the corresponding continuous cases, although now there are a few interesting differences.

2.3.2.3 Results for discrete distributions

The major difference between results for discrete and continuous uniform distributions is that for certain combinations of j and k, we have $\bar{W}_{\mathrm{BF}}^n(U\{j,k\}) = O(1)$. That is, the expected waste is bounded independent of n, something that is impossible in the continuous uniform case. Before discussing such cases, however, let us first see what happens for the discrete uniform analogues of the continuous uniform distribution $U[0,1]$.

THEOREM 2.25 [CCG91][CJS95]. For $k > 1$,

A. $\bar{W}_{\mathrm{FF}}^n(U\{k,k\}) = \Theta(\sqrt{nk})$.

B. $\bar{W}_{\mathrm{BF}}^n(U\{k,k\}) = O(\sqrt{n}\log k)$.

C. For any open online algorithm A, $\bar{W}_A^n(U\{k,k\}) = \Omega(\sqrt{n\log k})$.

Note that this means that for fixed k, both Best and First Fit have $\Theta(\sqrt{n})$ expected waste, something that is impossible for any open online algorithm under $U[0,1]$. However, if one allows k to grow with n, say as $n^{1/3}$, one obtains the bounds reported above for the $U[0,1]$ case, except for a small discrepancy in case (B). Indeed, the upper bound on $\bar{W}_{\mathrm{FF}}^n(U[0,1])$ in Theorem 2.22 was obtained from the proof of claim (A) in the above theorem by taking $k = \lceil n^{1/3} \rceil$. We conjecture that the bound in (B) can be replaced by $\Theta(\sqrt{n}\log^{3/4} k)$, thus eliminating the one discrepancy. We also conjecture that there exists an open online algorithm meeting the lower bound of (C), operating along much the same principles as the algorithm in the corresponding continuous result.

For $j = k-1$, the results of Theorem 2.25 continue to hold, since the only significant difference between $U\{k,k\}$ and $U\{k-1,k\}$ is the presence of items of size $k/k = 1$. Such an item fills a single bin with no waste and no impact on the remainder of the packing. Reducing j just one step further, however, gives rise to bounded expected waste results of the type mentioned above.

THEOREM 2.26 [CJS93][KRS96]. For all $k > 2$, $\bar{W}_{\mathrm{BF}}^n(U\{k-2,k\}) = O(1)$.

This was proved for $k \le 10$ by Coffman, Johnson, Shor, and Weber in [CJS93]. They modeled the packing process as a multi-dimensional Markov chain and used linear programming to derive for each individual distribution a potential function that could be used to show that the Markov chain was stable. Kenyon, Rabani, and Sinclair generalized this to arbitrary k in [KRS96] by deriving a simple potential function that works for all k (but needs significantly more sophisticated arguments to show that it works).

The Best Fit algorithm is well suited for a Markov chain approach, since the ordering of the bins in the packing is irrelevant as far as wasted space is concerned. Thus a packing can be represented by the numbers N_i of bins with gaps of size i/k, $0 \le i < k$. This not only facilitates Markov chain proofs, but also allows for much faster simulations of the algorithm. No similar advantage holds for First Fit, which helps explain why much less is known about First Fit. Moreover, limited experiments suggest that the analog of Theorem 2.26 does not

j \ k	5	6	7	8	9	10	11	12	13	14
3	B	B	B	B	B	B	B	B	B	B
4		B	B	B	B	B	B	B	B	B
5			B	B	B	B	B	B	B	B
6				B	B	B	B	B	B	B
7					B	B	B	B	B	B
8						B	L	B*	B*	B
9							B	L	L*	B*
10								B	L*	L*
11									B	L*
12										B

Table 2.3: Simulation results for $\bar{W}^n_{\text{BF}}(U\{j,k\})$ when $j < k-1$.

B = Bounded Waste; L = Linear Waste; * = Unproved

hold for First Fit, and indeed that $\bar{W}^n_{\text{FF}}(U\{k-2,k\})$ may still be $\Omega(n^{1/2})$ for large enough fixed k.

When we consider distributions $U\{j,k\}$ with $j \le k-3$, behavior can become much worse, even for Best Fit, and begins to mimic that for the continuous case, where $\bar{W}^n_{\text{BF}}(U[0,b])$ is apparently linear for all $b \in (0,1)$. We conjecture that for any such b there exists an integer K_b such that $\bar{W}^n_{\text{BF}}(U\{j,k\})$ is linear whenever $k > K_b$ and $|b - j/k| < \min\{(b/2,(1-b)/2\}$. Simulations reported in [CJS93] support this conjecture, with the conclusions summarized in Table 2.3. Most of the empirical conclusions in the table have now been rigorously proved, the exceptions being those entries marked by a "*". The results along the diagonal follow from Theorem 2.26, while the remaining bounded-waste results can be proved using individually-derived potential functions. (The linear-programming-based technique for generating these potential functions seems to run out of gas once $k \ge 15$ [CJS93].) The proofs that $\bar{W}^n_{\text{BF}}(U\{8,11\})$ and $\bar{W}^n_{\text{BF}}(U\{9,12\})$ are both linear rely on much more complicated Markov chain arguments that involve lengthy computer-aided numerical computations. To date no general approach has been found to prove such linear-waste conjectures.

To complete the picture for discrete uniform distributions, we note that for sufficiently small values of j, one can get bounded-waste results much more easily, and even extend them to First Fit. For instance, it is easy see that under both First and Best Fit, the expected waste is bounded for all $U\{j,k\}$ with $j \in \{1,2\}$. More generally, we have the following.

THEOREM 2.27 [CCG91].

A. $\bar{W}^n_{\text{BF}}(U\{j,k\}) = O(1)$ whenever $k \ge j(j+3)/2$.

B. $\bar{W}^n_{\text{FF}}(U\{j,k\}) = O(1)$ whenever $k \ge j^2$.

This completes our discussion of what is known about the performance of standard online algorithms like First and Best Fit under the discrete uniform distributions. The normal next question to ask is what is the *best* performance one can hope for from an online algorithm on such distributions. Given the lower bounds we saw in the continuous case, the answer is surprising.

THEOREM 2.28 [CCG91]. For any distribution $F = U\{j,k\}$ with $1 \le j < k-1$, there exists an open online algorithm A_F such that $\bar{W}^n_{A_F}(F) = O(1)$.

This theorem is an application of an earlier result of Courcoubetis and Weber [CW86a] [CW86b] which applies to all discrete distributions, and provides an explicit test by which the expected waste under any such distribution can be determined up to a constant factor. In general performing the test is an NP-hard problem, but it can be done efficiently in several important special cases (such as that of the distributions $U\{j,k\}$). In these situations, the Courcoubetis-Weber result can yield much tighter bounds than does the general result of Rhee and Talagrand [RT93b] [RT93c] mentioned in Section 2.3.2.2 that holds for all distributions, both continuous and discrete. Moreover, most real-world bin packing applications involve discrete rather than continuous distributions. Here is a sketch of what is involved in the Courcoubetis-Weber test.

A *discrete distribution* $F = (s_F, P_F)$ is specified by a finite set of item sizes $s_F = \{s_F(1),\dots,s_F(j)\}$, $0 < s_F(i) \le 1$ for $1 \le i \le j$, together with a list of probabilities $P_F = (P_F(1),\dots,P_F(j))$ where $\sum_{i=1}^{j} P_F(i) = 1$. Let us call an integer vector $(c_1,\dots,c_q)$ a *packing configuration* for F if c_i items of size $s_F(i), i = 1,\dots,j$, can all be packed into a bin, i.e., if $\sum_{i=1}^{j} c_i s_F(i) \le 1$. If this sum is equal to 1 (there is no wasted space), then $(c_1,\dots,c_j)$ is a *perfect* packing configuration. Let Λ_F be the convex cone in $\mathbf{R}^q$ spanned by all non-negative linear combinations of perfect packing configurations for F. The Courcoubetis-Weber test asks whether P_F, considered as a vector, lies in the convex cone Λ_F. The answer to the test not only determines the best possible expected waste for an online algorithm, but also characterizes the expected waste in an optimal packing, which turns out to be the same to within a constant factor.

THEOREM 2.29 [CW86a][CW86b]. For any discrete distribution F there exists an open online algorithm A_F with $\bar{W}^n_{A_F}(F) = \Theta(\bar{W}^n_{\mathrm{OPT}}(F))$. Moreover,

A. $\bar{W}^n_{\mathrm{OPT}}(F) = O(1)$ if P_F is in the interior of Λ_F.

B. $\bar{W}^n_{\mathrm{OPT}}(F) = \Theta(n^{1/2})$ if P_F is on the surface of Λ_F.

C. $\bar{W}^n_{\mathrm{OPT}}(F) = \Theta(n)$ if P_F is outside Λ_F.

For specific classes of distributions one can avoid the NP-hardness inherent in the general Courcoubetis-Weber result by proving combinatorial theorems about perfect packings that imply that one of the three cases always holds. This is what is done in [CCG91] to prove Theorem 2.28. The specific perfect packing theorem proved in [CCG91] is stronger than what is needed, and much harder to prove, but it is easier to state: *For positive integers k, j, and r, with $k \ge j$, the list L of rj items, r each of sizes 1 through j, can be packed perfectly into bins of size k if and only if the sum of the rj item sizes is a multiple of k.*

The specific online algorithms embodied in Theorem 2.29 are randomized, but can be efficiently derandomized when F is a discrete uniform distribution. In their randomized form, they maintain a separate pool of bins for each of the possible perfect packing configurations, and assign each new item randomly to an appropriate pool, using probabilities determined by a linear program over the current numbers of partially-packed bins of each type. As described, a separate algorithm is needed for each distribution, although in the case of the discrete uniform distribution a single algorithm can be constructed that has $O(1)$ expected waste for all such distributions [CCG91]. As with the more general but less-effective algorithm of Rhee and Talagrand [RT93b] [RT93c], this algorithm works by learning the distribution as it goes along. Also as with the Rhee and Talagrand algorithm, it is only of theoretical interest, a statement that holds for the specific algorithms A_F as well. For more practical results, let us now turn to the offline case.

2.3.3 OFFLINE ALGORITHMS

Given the impressive average-case performance of online algorithms, it is not clear whether offline algorithms have much of an advantage as far as expected performance is concerned. However, at least for continuous uniform distributions First and Best Fit Decreasing still outperform First and Best Fit in important ways. A first result (both in our presentation and historically) is the following.

THEOREM 2.30 [Kno81][Lue82]. For $A \in \{\text{FFD}, \text{BFD}\}$,

$$\bar{W}_A^n(U[0,1]) = \Theta(\sqrt{n})$$

Note that this is better than can be obtained by the best possible open online algorithm by a factor of $\sqrt{\log n}$.

The determination of the growth rate for $\bar{W}_{\text{FFD}}^n(U[0,1])$ was the first significant average-case result proved for bin packing. Frederickson in [Fre80] showed that it was $O(n^{2/3})$. Knödel [Kno81] and Lueker [Lue82] then independently improved this to the correct bound. Once again a simple algorithm was needed as an intermediary to make the analysis go through, and the main advantage that Knödel and Lueker had over Frederickson was a better choice of an intermediary. Perhaps the simplest choice for the intermediary is the algorithm *Match* (MA) given by Karp in [Kar82b]. This algorithm is the same as FFD except that as soon as a bin receives its second item it is closed. It is easy to show that $\text{MA}(L) \geq \max\{\text{FFD}(L), \text{BFD}(L)\}$ for all lists L. The analysis of Match when $F = U[0,1]$ easily reduces to computing the expected maximum excursion of a symmetric random walk above its original position and yields $\bar{W}_{\text{MA}}^n(U[0,1]) = \Theta(n^{1/2})$ thus implying the desired upper bound for both FFD and BFD. The corresponding lower bound follows from the fact that $\bar{W}_{\text{OPT}}^n(U[0,1]) = \Theta(n^{1/2})$, which itself is proved via an analysis of the expected value of the lower bound $\text{OPT}(L) \geq \max\{s(L), |\{a \in L : s(a) > 1/2\}|\}$ (see also [CL91, p. 122]).

It should be noted, as observed in [Kno81], that the above results hold for any distribution F that is symmetric about $1/2$, where a random variable X and its distribution function F are *symmetric about* a if $X - a$ and $a - X$ have the same distribution [CL91, p. 103].

For the distributions $U[0,b]$, $0 < b < 1$, the average-case advantage of Best and First Fit Decreasing over their online counterparts increases substantially, although to date this has only been proved for FFD, as shown by Bentley, Johnson, Leighton, McGeoch, and McGeoch [BJL84].

THEOREM 2.31 [BJL84].

$$\bar{W}^n_{\mathrm{FFD}}(U[0,b]) = \begin{cases} \Theta(1) & \text{if } b \le 1/2 \\ \Theta(n^{1/3}) & \text{if } 1/2 < b < 1. \end{cases}$$

These results should be compared to our earlier result that any open online algorithm must have expected waste at least $\Omega(\sqrt{n})$ for $0 < b < 1$, and our observation that both First and Best Fit appear to have linear expected waste for all such b.

When $b \le 1/2$ the constant bound appears to be no more than 0.7 based on simulations reported in [BJL83], meaning that FFD is typically optimal or at most one bin off. The proof in [BJL84] unfortunately only provided a bound of something like 10^{10}. A later analysis of the case $0 < b \le 1/2$ by Floyd and Karp [FK91] reduced the bound on the constant hidden in the $\Theta(1)$ notation to 11.3, albeit at a slight change in the probabilistic model. Their technique was to take the number of items in L to be Poisson distributed with mean n, and then to reduce the problem to the analysis of a queueing system; the discontinuity from bounded to unbounded expected wasted space when $b = 1/2$ was then explained by its correspondence with the stability point of the queueing system.

As to the situation when $b > 1/2$, it is claimed in [BJL84] that one can do substantially better than FFD by modifying the algorithm slightly.

THEOREM 2.32 [BJL84]. There exists an $O(n \log n)$-time offline algorithm A such that for $1/2 < b < 1$, $\bar{W}^n_A(U[0,b]) = O(1)$.

It should be noted that the proofs of Theorem 2.32 and the second half of Theorem 2.31 currently still exist only in the form of handwritten notes, and so we may have been generous in calling these results "theorems" at this point.

When we turn to discrete uniform distributions, the advantages of FFD and BFD are no longer so clear in light of Theorem 2.29, which says for any discrete distribution there is an online algorithm with the same expected waste as an optimal packing to within a constant factor. Moreover, although as one might expect these two algorithms can outperform First and Best Fit, they do not do so consistently. For instance, for $F = U\{8,11\}$ we have $\bar{W}^n_{\mathrm{BFD}}(F) = O(1)$ and $\bar{W}^n_{\mathrm{BF}}(F) = \Theta(n)$, while for $F = U\{6,13\}$ we have $\bar{W}^n_{\mathrm{BFD}}(F) = \Theta(n)$ and $\bar{W}^n_{\mathrm{BF}}(F) = O(1)$. These results for BFD are based on a *fluid packing* theorem mentioned in [CCG91] and proved in [CJM96].

In the fluid packing of a discrete distribution $F = (s_F, P_f)$ by First or Best Fit Decreasing, one views the distribution as made up of continuous quantities of each item-size, with an amount $P_F(i)$ of item-size $s_F(i)$. One then performs a version of the algorithm that treats each item-size in turn (in decreasing order), building up a packing that contains fractional amounts of the resulting bin-types. For example, when $F = U\{6,13\}$, FFD begins by using up the given amount $1/6$ of item-size $6/13$ to create an amount $1/12$ of the bin-type that consists of two items of size $6/13$. When it comes time to pack item-size $1/13$, half of the amount of this item-size will fill the gaps in these "bins." The final packing resulting from this fluid packing process is called *perfect* if all the bin types occurring in the final packing in nonzero quantities contain no wasted space. Coffman, Johnson, McGeoch, Shor, and Weber [CJM96] prove the following general result.

THEOREM 2.33 [CJM96]. Let $F = (s_F, P_f)$ be a discrete distribution with j item sizes and let $A \in \{\mathrm{FFD}, \mathrm{BFD}\}$. Then

- ***A.*** If the fluid packing of A for F is not perfect, then $\bar{W}_A^n(F) = \Theta(n)$.
- ***B.*** If the fluid packing of A for F is perfect, then $\bar{W}_A^n(F)$ is either $O(1)$ or $\Theta(\sqrt{n})$. The first option occurs if and only if there exists a $\delta > 0$ such that for all probability distributions Q on S_F with $\max\{|Q(i) - P_F(i)| : 1 \le i \le j\} < \delta$ the fluid packing of A for $F' = (s_F, Q)$ is perfect.

The theorem actually holds for a more general class of offline algorithms called *bin type priority algorithms* in [CJM96]. The condition stated in (B) can be tested by solving a finite number of linear programs, but in general is NP-hard even for BFD and FFD [CJM96]. In the case of the discrete uniform distribution $U\{j,k\}$, however, it can be tested in time $O(j \log j)$.

The above results are similar in form to those we saw in Theorem 2.29 for the best possible online algorithms when F is discrete. The consequences are not nearly so simple, however. For $j \in \{k-1, k\}$ and $k > 3$ the expected waste is always $\Theta(\sqrt{n})$, although now the constant of proportionality need not depend on k, as can be proved using arguments like those used in the continuous case for $U[0,1]$. Using the fluid packing test, the nature of $\bar{W}_F^n(F)$ when $j < k-1$ has been determined for both BFD and FFD for all $U\{j,k\}$ with $k \le 1,000$ [CCG91] (and in the case of FFD for all $k \le 2,500$ [CJM96]). All cases examined so far yield expected waste $O(1)$ or $\Theta(n)$ with the same answer for both BFD and FFD, but the patterns of pairs (j,k) for which these two options occur is not at all straightforward. For a given k the values of j for which linear waste occurs are often broken up into multiple intervals separated by j's for which the expected waste is $O(1)$, and k's exist with as many as 10 such intervals. See [CCG91] for more details.

Certain global observations are possible, however. When $\bar{W}_A^\infty(U\{j,k\}) = \Theta(n)$ for $A \in \{\mathrm{FFD}, \mathrm{BFD}\}$, the fluid packing process can be used to determine the precise value of $\bar{R}_A^\infty(U\{j,k\})$, and based on the computations for $k \le 2,500$ and additional arguments, one can show the following.

THEOREM 2.34 [CCG91][CJM96]. For $A \in \{\mathrm{FFD}, \mathrm{BFD}\}$ and all pairs (j,k) with $j < k-1$,

A. $\bar{R}_A^\infty(U\{j,k\}) \le \bar{R}_A^\infty(U\{6/13\}) \approx 1.00596$.
B. $\bar{R}_A^\infty(U\{j,k\}) = O(\log k/k)$.

Thus the asymptotic expected ratios are bounded and go to 0 as $K \to \infty$. The proof uses a simplified version of FFD as an intermediary that shares the name *Modified First Fit Decreasing* and abbreviation MFFD with the quite different algorithm of [GJ85] discussed in Section 2.2.8. In this version of MFFD a bin is closed as soon as it receives its first *fallback item*, where a fallback item is an item that is added to a bin after a subsequent bin has been started. (This version of MFFD is also used by Floyd and Karp [FK91] in their analysis of FFD for $U[0,b]$ when $b \le 1/2$.)

This concludes our coverage of First and Best Fit Decreasing. The only other offline algorithm whose average case behavior has received serious attention is Next Fit Decreasing (NFD), with the attention probably due more to the ease with which it can be analyzed than to any likely applications for the algorithm. The average-case analysis of NFD began with Hofri and Kamhi [HK86] (see also Hofri [Hof87]) and continued soon after with Csirik et al. [CFF86]. These papers contain several results including the fact that $\bar{R}_{\mathrm{NFD}}^\infty(U[0,1]) = \pi^2/3 - 2 \approx 1.2899$ (the same value we saw in Theorem 2.18 for $\lim_{K\to\infty} \bar{R}_{\mathrm{H}_K}^\infty(U[0,1])$, for much the same reason). This result is easily generalized to arbitrary distributions F [Rhe87]. With F general, let α_k be the probability that an item size falls in $(1/(k+1), 1/k]$, and define $\sigma_F = \sum_{k\ge1} \alpha_k/k$. Rhee shows that $\bar{R}_{\mathrm{NFD}}^\infty(F) = \sigma_F$

2.3.4 OTHER AVERAGE-CASE QUESTIONS

In this section we consider two additional questions about average-case behavior that bear on the results presented above. The first concerns the quality of optimal packings.

2.3.4.1 When is the expected optimal packing "perfect"?

A distribution F is said to *allow perfect packings* if $\bar{W}_{\mathrm{OPT}}^n(F) = o(n)$. Note that this implies that $\lim_{n\to\infty} E[\mathrm{OPT}(L_n)/s(L_n)] = 1$ under F, since $E[s(L_n)] = \mu n$, where μ is the mean of F. The question of which distributions allow perfect packings, originally posed by Karp as reported in [Kar82a], has been a subject of much study. It is germane to our concerns about the performance of heuristics, since for those F that allow perfect packing, we can simplify our estimates of $\bar{R}_A^\infty(F)$ by simply comparing values of $E[A(L_n)]$ to μn.

Fortunately for our analysis, all the discrete and continuous uniform distributions that we have been examining do allow perfect packings, as is implied by the results already presented. On the other hand, perhaps our understanding of algorithmic performance would be more thorough if it contained some knowledge about average-case results for distributions that do not allow perfect packings. Be that as it may, the distributions on which we have concentrated in fact allow even more "perfect" packings than the definition requires. For all the continuous uniform distributions $F = U[0,b]$, $0 < b < 1$, and all the discrete uniform distributions $F = U\{j,k\}$, $k > 1$ and $j \le k-2$, we have $\bar{W}^n_{\mathrm{OPT}}(F) = O(1)$ as a consequence of Theorems 2.28, 2.31, and 2.32. For the remaining cases, $F = U[0,1]$ and $F \in \{U\{k-1,k\}, U\{k,k\} : k \ge 3\}$, we have $\bar{W}^n_{\mathrm{OPT}}(F) = \Theta(\sqrt{n})$.

For the case of $U[0,1]$, even more detailed information is known, as various authors have attempted to estimate the multiplicative constant of the $\sqrt{n}$ term. The current best estimate, due to Csirik, Frenk, Galambos, and Rinnooy Kan [CFG91], is that

$$\bar{W}^n_{\mathrm{OPT}}(U[0,1]) = \sqrt{\frac{n}{32\pi}} + o(\sqrt{n}).$$

In contrast, most searches for distributions that allow perfect packings have been willing to settle for simply showing that the expected waste is $o(n)$, which is all that the definition requires, or $O(\sqrt{n})$, which is typically the case. Although the original proof that $U[0,1]$ allows perfect packings was implicit in the 1980 analysis of FFD by Frederickson [Fre80], the first explicit study of the question appears to be that of Karmarkar [Kar82a]. Implicit in Frederickson's result was the fact that any symmetric distribution (as defined in the previous section) allows perfect packings. Karmarkar extended this to show that the same held for any *decreasing* distribution, i.e., one whose density function is non-increasing. The proof is based on the fact that any such distribution can be decomposed into the union of a sequence of distributions that are symmetric around powers of $1/2$, an observation also made by Loulou [Lou84a] and partially attributable to Knödel [Kno81].

When Karp originally asked the question about perfect packings, he was concerned with the class of continuous uniform distributions $U[a,b]$ with $0 < a < b \le 1$. For this class there exist examples of distributions that trivially do not allow perfect packings, for instance all those with $a > 1/2$ or $a \le 1/2$ and $b > 1-a$, as well as others with more subtle reasons for being bad. Lueker in [Lue83] identified all of these using a proof technique motivated by linear programming duality. He also showed that $U[a,b]$ allows perfect packings whenever $(a+b)/2$ is the reciprocal of an integer $m \ge 3$, although he was unable to show that all the distributions outside of his bad class allowed perfect packings.

Rhee [Rhe88, Rhe90] and Rhee and Talagrand [RT88b, RT89d, RT89e] continued this research, addressing the general problem of characterizing *all* distributions on $[0,1]$ that allow perfect packing. Incorporating results in topology and functional analysis, they developed a comprehensive and deep theory. We present here a few of the major results, starting with a fundamental characterization of the set $\mathcal{B}$ of measures allowing perfect packing. Let $R_k, k \ge 1$, be the set of k-tuples $(x_1, \ldots, x_k)$ such that $0 \le x_i \le k$ and $\sum_{i=1}^{k} x_i = 1$. Let M_k denote the set of all probability measures on R_k, and let $\mathcal{B}_k$ denote the set of all probability measures $\hat{\nu}$ on $[0,1]$ induced by the measure $\nu \in M_k$ as follows: Sample R_k according to ν, then choose a single component of the sample, with all k components equally likely. Rhee [Rhe88] proved that $\mathcal{B}$ is the class of all probability measures obtainable as (countable) positive linear combinations of measures chosen from the sets $\mathcal{B}_k$. Following up on this work, Rhee and Talagrand [RT88b] derived more explicit sufficient conditions for F to allow perfect packings, and as a corollary resolved the questions left open by Lueker's earlier work on the uniform distributions, thus completing the solution to Karp's perfect packing problem.

In general, the proofs that various classes of distributions allow perfect packing have not been constructive, i.e., optimal packing algorithms have not been given. Exceptions are the cases mentioned above of decreasing distributions and distributions $U[a,b]$ with $(a+b)/2 = 1/m$, as well the case of any triangular density whose expectation is $1/m$ with $m \geq 3$ an integer, for which Krause, Larmore, and Volper [KLV87] have given a constructive proof that perfect packings are allowed.

2.3.4.2 What can we say about the probability distribution of $A(L_n)$?

It is one thing to know the expected number of bins $A(L_n)$ that an algorithm A will produce given a list L_n generated under distribution F. In many applications, however, it is useful to know more about the distribution of $A(L_n)$, such as the nature of its tails. This might have practical significance in the computation of safety margins, but it also has theoretical significance for our analysis. Although we have defined $\bar{R}_A^n(F)$ in terms of the expected ratio $E[A(L_n)/\mathrm{OPT}(L_n)]$, it is often much easier to prove results about $E[A(L_n)]/E[\mathrm{OPT}(L_n)]$, the ratio of expectations. Fortunately, for any distribution F the tails of the distribution of $\mathrm{OPT}(L_n)$ decline sufficiently rapidly with n that these ratios must converge to the same limit as $n \to \infty$, and this is all we are interested in when we talk about $\bar{R}_A^\infty(F)$.

To be specific, consider the following result of Rhee and Talagrand [RT87]. Recall from Section 2.2.10.2 that we call an algorithm A *monotone* if an increase in the sizes or numbers of items in L never causes $A(L)$ to decrease for any list L. Say an algorithm is *k-conservative* if inserting a new item in L never increases $A(L)$ by more than k. Next Fit is one the few of our standard heuristics that meet both these constraints (the latter with $k = 2$). However, OPT, viewed as an algorithm, also meets these criteria, in this case with $k = 1$. Using martingale techniques, Rhee and Talagrand proved the following result.

THEOREM 2.35 [RT87]. For any monotone, k-conservative algorithm A and any distribution F,

$$P(|A(L_n)) - E[A(L_n)]| \geq t) \leq 2e^{-\alpha t^2/n}$$

where $\alpha > 0$ is a constant depending on the value of k but not on F.

For OPT we have $\alpha = 1/2$ and for NF we have $\alpha = 1/8$. Thus, if we take $t = \sqrt{4n \ln n}$, we get that the probability that $\mathrm{OPT}(L_n)$ exceeds its mean by more than t is no more than $2/n^2$. Given that no algorithm A can have a worst-case ratio $R_A^n(F) > n$, it is then an easy exercise to show that $E[A(L_n)/\mathrm{OPT}(L_n)]$ and $E[A(L_n)]/E[\mathrm{OPT}(L_n)]$ go to the same limit as $n \to \infty$.

In addition to Theorem 2.35, there have been many other results giving similar exponential type bound on tail probabilities. These include ones that tighten up Theorem 2.35 for particular distributions F [Rhe89] [Rhe93a] [Rhe94], as well as weaker results covering algorithms not captured by Theorem 2.35, such as MFFD (the version discussed in Section 2.3.3) [Rhe91], BFD and FFD [RT89b] [RT89c], and stronger results for algorithms like Next Fit Decreasing that were captured by the theorem [Rhe87]. See also [Rhe85] [Rhe89] [Rhe93b] [RT89a] [CL91].

CONCLUSION 2.4

In this chapter we have tried to give a relatively complete picture of the state of the art with respect to the classical one-dimensional bin packing problem, both from the worst-case point of view that is the common outlook of this book and from an average-case point of view that helps to put those worst-case results into perspective. The field of bin packing is much wider than the single classical problem studied here, however, and many of the same questions have been studied in great detail for a wide range of variants. Among these are generalizations to higher dimensions [CR89], variants in which additional constraints are present, such as precedence relations [WM82] or bounds on the maximum number of items allowed in a bin [KSS75], variants in which bins of different sizes are allowed [FL86a], variants in which items have lifetimes and may leave the packing before subsequent items arive [CGJ83] variants in which each non-empty bin must contain items of total size *at least* a given amount and we wish to maximize the number of bins used [AJK84], and variants in which the number of bins is fixed and some other objective function is considered, such as minimizing the maximum bin contents (the multiprocessor scheduling problem [Gra69]).

The literature on such generalizations is vast, and the references given above should only be viewed as existence proofs for work on the problems in question. The current authors' earlier survey [CGJ84] elaborates on the variety of problems and covers the literature through 1984. There are many important references that have appeared since it was written, however, especially in the multidimensional cases, and an expanded version of this chapter is planned that will bring the entire picture up to date [CGJ97].

References

[AJK84] S. B. Assman, D. S. Johnson, D. J. Kleitman, and J. Y-T. Leung. On a dual version of the one-dimensional bin packing problem. *J. Algorithms*, 5:502–525, 1984.

[AKT84] M. Ajtai, J. Komlós, and G. Tusnády. On optimal matchings. *Combinatorica*, 4:259–264, 1984.

[AMW89] R. J. Anderson, E. W. Mayr, and M. K. Warmuth. Parallel approximation algorithms for bin packing. *Inf. and Comput.*, 82:262–277, 1989.

[Bak83] B. S. Baker. A new proof for the first-fit decreasing bin-packing algorithm. *J. Algorithms*, 6:49–70, 1985.

[BC81] B. S. Baker and E. G. Coffman, Jr. A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM J. Alg. Disc. Meth.*, 2:147–152, 1981.

[BE83] J. Blazewicz and K. Ecker. A linear time algorithm for restricted bin packing and scheduling problems. *Oper. Res. Lett.*, 2:80–83, 1983.

[BJL83] J. L. Bentley, D. S. Johnson, F. T. Leighton, and C. C. McGeoch. An experimental study of bin packing. In *Proceedings of the 21st Annual Allerton Conference on Communication, Control, and Computing*, pages 51–60, Urbana, 1983. University of Illinois.

[BJL84] J. L. Bentley, D. S. Johnson, F. T. Leighton, C. C. McGeoch, and L. A. McGeoch. Some unexpected expected behavior results for bin packing. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 279–288, 1984.

[Bro71] A. R. Brown. *Optimum Packing and Depletion.* American Elsevier, New York, 1971.

[Bro79] D. J. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1979.

[CCG91] E. G. Coffman, Jr., C. A. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeogh, P. W. Shor, R. R. Weber, and M. Yannakakis. Fundamental discrepancies between average-case analyses under discrete and continuous distributions: A bin packing case study. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 230–240. ACM Press, 1991.

[CFF86] J. Csirik, J. B. G. Frenk, A. Frieze, G. Galambos, and A. H. G. Rinnooy Kan. A probabilistic analysis of the next fit decreasing bin packing heuristic. *Oper. Res. Lett.*, 5:233–236, 1986.

[CFG91] J. Csirik, J. B. G. Frenk, G. Galambos, and A. H. G. Rinnooy Kan. Probabilistic analysis of algorithms for dual bin packing problems. *J. Algorithms*, 12:189–203, 1991.

[CG86] J. Csirik and G. Galambos. An $O(n)$ bin-packing algorithm for uniformly distributed data. *Computing*, 36:313–319, 1986.

[CGJ83] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12:227–258, 1983.

[CGJ84] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin-packing: An updated survey. In G. Ausiello, M. Lucertini, and P. Serafini, editors, *Algorithm Design for Computer System Design*, pages 49–106. Springer-Verlag, Wien, 1984. CISM Courses and Lectures Number 284.

[CGJ87] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Bin packing with divisible item sizes. *J. Complexity*, 3:405–428, 1987.

[CGJ97] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. To appear.

[Cha92] B. Chandra. Does randomization help in on-line bin packing? *Information*

Proc. Lett., 43:15–19, 1992.

[CI89] J. Csirik and B. Imreh. On the worst-case performance of the NkF bin-packing heuristic. *Acta Cybernetica*, 9:89–105, 1989.

[CJ91] J. Csirik and D. S. Johnson. Bounded space on-line bin packing: Best is better than first. In *Proceedings, Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 309–319, Philadelphia, 1991. Society for Industrial and Applied Mathematics.

[CJ92] J. Csirik and D. S. Johnson. Bounded space on-line bin packing: Best is better than first. *Algorithmica*, submitted.

[CJM96] E. G. Coffman, Jr., D. S. Johnson, L. A. McGeoch, P. W. Shor, and R. R. Weber. Bin packing with discrete item sizes, part III: Average-case behavior of FFD and BFD, 1996. In preparation.

[CJS93] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber. Markov chains, computer proofs, and best fit bin packing. In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 412–421, New York, 1993. ACM Press.

[CJS95] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber. Bin packing with discrete item sizes, part II: Average-case behavior of first fit, 1995. manuscript.

[CL91] E. G. Coffman, Jr. and G. S. Lueker. *Probabilistic Analysis of Packing and Partitioning Algorithms*. Wiley, New York, 1991.

[CR89] D. Coppersmith and P. Raghavan. Multidimensional on-line bin packing: Algorithms and worst-case analysis. *Oper. Res. Lett.*, 8:17–20, 1989.

[CS91] E. G. Coffman, Jr. and P. W. Shor. A simple proof of the $O(\sqrt{n}\log^{3/4} n)$ up-right matching bound. *SIAM J. Disc. Math.*, 4:48–57, 1991.

[CS93] E. G. Coffman, Jr. and P. W. Shor. Packing in two dimensions: Asymptotic average-case analysis of algorithms. *Algorithmica*, 9:253–277, 1993.

[CSB94] L. M. A. Chan, D. Simchi-Levi, and J. Bramel. Worst-case analyses, linear programming, and the bin-packing problem, 1994. Manuscript.

[CSH80] E. G. Coffman, Jr., K. So, M. Hofri, and A. C. Yao. A stochastic model of bin-packing. *Inf. and Cont.*, 44:105–115, 1980.

[Csi93] J. Csirik. The parametric behavior of the first-fit decreasing bin packing algorithm. *J. Algorithms*, 15:1–28, 1993.

[CW86a] C. Courcoubetis and R. R. Weber. A bin-packing system for objects with sizes from a finite set: Necessary and sufficient conditions for stability and some applications. In *Proceedings of the 25th IEEE Conference on Decision and Control*, pages 1686–1691, Athens, Greece, 1986.

[CW86b] C. Courcoubetis and R. R. Weber. Necessary and sufficient conditions for stability of a bin packing system. *J. Appl. Prob.*, 23:989–999, 1986.

[FK91] S. Floyd and R. M. Karp. FFD bin packing for item sizes with distributions on $[0,1/2]$. *Algorithmica*, 6:222–240, 1991.

[FL81] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[FL86a] D. K. Friesen and M. A. Langston. Variable sized bin packing. *SIAM J.*

Comput., 15:222–230, 1986.

[FL91] D. K. Friesen and M. A. Langston. Analysis of a compound bin-packing algorithm. *SIAM J. Disc. Math*, 4:61–79, 1991.

[Fre80] G. N. Frederickson. Probabilistic analysis for simple one- and two-dimensional bin packing algorithms. *Inf. Proc. Lett.*, 11:156–161, 1980.

[Gal86] G. Galambos. Parametric lower bound for on-line bin-packing. *SIAM J. Alg. Disc. Meth.*, 7:362–367, 1986.

[GF93] G. Galambos and J. B. G. Frenk. A simple proof of Liang's lower bound for on-line bin packing and the extension to the parametric case. *Disc. Appl. Math.*, 41:173–178, 1993.

[GG61] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Oper. Res.*, 9:948–859, 1961.

[GG63] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock program — Part II. *Oper. Res.*, 11:863–888, 1963.

[GGJ76] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. Yao. Resource constrained scheduling as generalized bin packing. *J. Comb. Th. Ser. A*, 21:257–298, 1976.

[GGU71] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings, 4th Annual Symposium on Theory of Computing*, pages 143–150, New York, 1972. ACM.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.

[GJ81] M. R. Garey and D. S. Johnson. Approximation algorithms for bin-packing problems: A survey. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms in Combinatorial Optimization*, pages 147–172. Springer-Verlag, New York, 1981.

[GJ85] M. R. Garey and D. S. Johnson. A 71/60 theorem for bin packing. *J. of Complexity*, 1:65–106, 1985.

[GPT90] G. Gambosi, A. Postiglione, and M. Talamo. New algorithms for on-line bin packing. In R. Petreschi G. Aussiello, D. P. Bovet, editor, *Algorithms and Complexity, Proceedings of the First Italian Conference*, pages 44–59, Singapore, 1990. World Scientific.

[Gra69] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17:263–269, 1969.

[Gra72] R. L. Graham. Bounds on multiprocessing anomalies and related packing algorithms. In *Proc. 1972 Spring Joint Computer Conference*, pages 205–217, Montvale, NJ, 1972. AFIPS Press.

[Gro94] E. F. Grove. Online bin packing with lookahead. Unpublished manuscript, July 1994.

[GW93b] G. Galambos and G. J. Woeginger. Repacking helps in bounded space on-line bin-packing. *Computing*, 49:329–338, 1993.

[Hal74] S. Halász. Private communication, 1974.

[Hal89] S. Halfin. Next-fit bin packing with random piece sizes. *J. Appl. Prob.*, 26:503–511, 1989.

[HK86] M. Hofri and S. Kamhi. A stochastic analysis of the NFD bin-packing algorithm. *J. Algorithms*, 7:489–509, 1986.

[HK88] T. C. Hu and A. B. Kahng. Anatomy of on-line bin packing. Technical Report CSE-137, Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA, 1988.

[Hof82] U. Hoffman. A class of simple stochastic online bin packing algorithms. *Computing*, 29:227–239, 1982.

[Hof87] M. Hofri. *Probabilistic Analysis of Algorithms.* Springer-Verlag, New York, 1987.

[IL93] Z. Ivković and E. Lloyd. Fully dynamic algorithms for bin packing: Being myopic helps. In *Proceedings of the First European Symposium on Algorithms*, number No. 726 in Lecture Notes in Computer Science, pages 224–235, New York, 1993. Springer Verlag. Journal version to appear in *SIAM J. Comput.*

[IL94] Z. Ivkovič and E. Lloyd. Partially dynamic bin packing can be solved within $1+\epsilon$ in (amortized) polylogarithmic time. Technical report, Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, 1994.

[JDU74] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3:299–325, 1974.

[JKS95] D. S. Johnson, C. Kenyon, P. W. Shor, and N. Young, 1995. Private communication.

[Joh73] D. S. Johnson. *Near-Optimal Bin Packing Algorithms.* PhD thesis, Massachusetts Institute of Technology, Department of Mathematics, Cambridge, 1973.

[Joh74] D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.

[Joh82] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. Algorithms*, 3:288–300, 1982.

[Joh96] D. S. Johnson, 1996. Unpublished results.

[Kar82a] N. Karmarkar. Probabilistic analysis of some bin-packing algorithms. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 107–111, 1982.

[Kar82b] R. M. Karp. Lecture notes. Computer Science Division, University of California, Berkeley, 1982.

[Ken96] C. Kenyon. Best-fit bin-packing with random order. In *Proceedings, The Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, Philadelphia, 1996. Society for Industrial and Applied Mathematics.

[KK82] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proc. 23rd Ann. Symp. on Foundations of Computer Science*, pages 312–320, 1982. IEEE Computer Soc.

[KLM84b] R. M. Karp, M. Luby, and A. Marchetti-Spaccamela. A probabilistic analysis of multidimensional bin packing problems. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 289–298, 1984.

[KLV87] K. Krause, L. Larmore, and D. Volper. Packing items from a triangular distribution. *Inf. Proc. Lett.*, 25:351–361, 1987.

[Kno81] W. Knödel. A bin packing algorithm with complexity $O(n \log n)$ and performance 1 in the stochastic limit. In J. Gruska and M. Chytil, editors, *Proceedings 10th Symp. on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, 118, pages 369–378, Berlin, 1981. Springer-Verlag.

[KRS96] C. Kenyon, Y. Rabani, and A. Sinclair. Biased random walks, Lyapunov functions, and stochastic analysis of best fit bin packing. In *Proc. Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 351–358, Philadelphia, 1996. Society for Industrial and Applied Mathematics.

[KSS75] K. L. Krause, Y. Y. Shen, and H. D. Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *J. Assoc. Comput. Mach.*, 22:522–550, 1975.

[Len83] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[Lia80] F. M. Liang. A lower bound for on-line bin packing. *Inf. Proc. Lett.*, 10:76–79, 1980.

[LL85] C. C. Lee and D. T. Lee. A simple on-line packing algorithm. *J. ACM*, 32:562–572, 1985.

[LL87] C. C. Lee and D. T. Lee. Robust on-line bin packing algorithms. Technical report, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 1987.

[Lou84a] R. Loulou. Probabilistic behavior of optimal bin-packing solutions. *Oper. Res. Lett.*, 3:129–135, 1984.

[LS89] T. Leighton and P. Shor. Tight bounds for minimax grid matching with applications to the average case analysis of algorithms. *Combinatorica*, 9:161–187, 1989.

[Lue82] G. S. Lueker. An average-case analysis of bin packing with uniformly distributed item sizes. Technical Report 181, University of California at Irvine, Department of Information and Computer Science, 1982.

[Lue83] G. S. Lueker. Bin packing with items uniformly distributed over intervals $[a, b]$. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, pages 289–297, 1983.

[Mao93a] W. Mao. Best-k-fit bin packing. *Computing*, 50:265–270, 1993.

[Mao93b] W. Mao. Tight worst-case performance bounds for next-k-fit bin packing. *SIAM J. Comput.*, 22:46–56, 1993.

[Mar85] C. U. Martel. A linear time bin-packing algorithm. *Oper. Res. Lett.*, 4:189–192, 1985.

[Mur85] F. D. Murgolo. *Approximation Algorithms for Combinatorial Optimization Problems*. PhD thesis, Univesity of California at Irvine, 1985.

[Mur88] F. D. Murgolo. Anomalous behavior in bin packing algorithms. *Disc. Appl. Math.*, 21:229–243, 1988.

[OMW84] H. L. Ong, M. J. Magazine, and T. S. Wee. Probabilistic analysis of bin packing heuristics. *Oper. Res.*, 32:993–998, 1984.

[Ram89] P. Ramanan. Average-case analysis of the smart next fit algorithm. *Inf. Proc. Lett.*, 31:221–225, June 12 1989.

[RBL89] P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. On-line bin packing in linear time. *J. Algorithms*, 10:305–326, 1989.

[Rhe85] W. T. Rhee. Convergence of optimal stochastic bin packing. *Oper. Res. Lett.*, 4:121–123, 1985.

[Rhe87] W. T. Rhee. Probabilistic analysis of the next fit decreasing algorithm for bin packing. *Oper. Res. Lett.*, 6:189–191, 1987. Correction: *Oper. Res. Lett.*, 7:211, 1988.

[Rhe88] W. T. Rhee. Optimal bin packing with items of random sizes. *Math. Oper. Res.*, 13:140–151, 1988.

[Rhe89] W. T. Rhee. Some inequalities for bin packing. *Optimization*, 20:299–304, 1989.

[Rhe90] W. T. Rhee. A note on optimal bin packing and optimal bin covering with items of random size. *SIAM J. Comput.*, 19:705–710, 1990.

[Rhe91] W. T. Rhee. Stochastic analysis of a modified first fit decreasing packing. *Math. Oper. Res.*, 16:162–175, 1991.

[Rhe93a] W. T. Rhee. Optimal bin packing of items of sizes uniformly distributed over $[0,1]$. *Math. Oper. Res.*, 18:694–704, 1993.

[Rhe93b] W. T. Rhee. Inequalities for Bin Packing – II. *Math. Oper. Res.*, 18:685–693, 1993.

[Rhe94] W. T. Rhee. Inequalities for Bin Packing – III. *Optimization*, 29:381–385, 1994.

[Ric91] M. B. Richey. Improved bounds for harmonic-based bin packing algorithms. *Disc. Appl. Math.*, 34:203–227, 1991.

[RT87] W. T. Rhee and M. Talagrand. Martingale inequalities and NP-complete problems. *Math. Oper. Res.*, 12:177–181, 1987.

[RT88a] W. T. Rhee and M. Talagrand. Exact bounds for the stochastic upward matching problem. *Trans. Amer. Math. Soc.*, 307:109–125, 1988.

[RT88b] W. T. Rhee and M. Talagrand. Some distributions that allow perfect packing. *Assoc. Comp. Mach.*, 35:564–578, 1988.

[RT89a] W. T. Rhee and M. Talagrand. The complete convergence of best fit decreasing. *SIAM J. Comput.*, 18:909–918, 1989.

[RT89b] W. T. Rhee and M. Talagrand. The complete convergence of first fit decreasing. *SIAM J. Comput.*, 18:919–938, 1989.

[RT89c] W. T. Rhee and M. Talagrand. Optimal bin covering with items of random size. *SIAM J. Comput.*, 18:487–498, 1989.

[RT89d] W. T. Rhee and M. Talagrand. Optimal bin packing with items of random sizes—II. *SIAM J. Comput.*, 18:139–151, 1989.

[RT89e] W. T. Rhee and M. Talagrand. Optimal bin packing with items of random sizes – III. *SIAM J. Comput.*, 18:473–486, 1989.

[RT89f] P. Ramanan and K. Tsuga. Average-case analysis of the modified harmonic algorithm. *Algorithmica*, 4:519–533, 1989.

[RT93b] W. T. Rhee and M. Talagrand. On line bin packing with items of random size. *Math. Oper. Res.*, 18:438–445, 1993.

[RT93c] W. T. Rhee and M. Talagrand. On line bin packing with items of random sizes – II. *SIAM J. Comput.*, 22:1251–1256, 1993.

[Sha77] S. D. Shapiro. Performance of heuristic bin packing algorithms with segments of random length. *Inf. and Cont.*, 35:146–148, 1977.

[Sho86] P. W. Shor. The average-case analysis of some on-line algorithms for bin packing. *Combinatorica*, 6(2):179–200, 1986.

[Sho91] P. W. Shor. How to pack better than best-fit: Tight bounds for average-case on-line bin packing. In *Proceedings, 32nd Annual Symposium on Foundations of Computer Science*, pages 752–759, New York, 1991. IEEE Computer Society Press.

[Sim94] D. Simchi-Levi. New worst-case results for the bin packing problem. *Nav. Res. Log.*, 41:579–585, 1994.

[Tal94] M. Talagrand. Matching theorems and empirical discrepancy computations using majorizing measures. *J. Amer. Math. Soc.*, 7:455–537, 1994.

[Ull71] J. D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, October 1971.

[Vli95] A. van Vliet. *Lower and Upper Bounds for On-Line Bin Packing and Scheduling Heuristic*. PhD thesis, Erasmus University, Rotterdam, Netherlands, 1995.

[Vli96] A. van Vliet. On the asymptotic worst case behavior of harmonic fit. *J. Algorithms*, 20:113–136, 1996.

[WM82] T. S. Wee and M. J. Magazine. Assembly line balancing as generalized bin-packing. *Oper. Res. Lett.*, 1:56–58, 1982.

[Woe93] G. Woeginger. Improved space for bounded-space, on-line bin-packing. *SIAM J. Disc. Math.*, 6:575–581, 1993.

[Xu93] K. Xu. *A Bin-Packing Problem with Item Sizes in the Interval* $(0, \alpha]$ *for* $\alpha \leq \frac{1}{2}$. PhD thesis, Chinese Academy of Sciences, Institute of Applied Mathematics, Beijing, China, 1993.

[Yao80] A. C. Yao. New algorithms for bin packing. *J. Assoc. Comput. Mach.*, 27:207–227, 1980.

[Yue91] M. Yue. A simple proof of the inequality $\mathrm{FFD}(L) \leq \frac{11}{9}\mathrm{OPT}(L) + 1\ \forall L$, for the FFD bin-packing algorithm. *Acta Math. App. Sinica*, 7:321–331, 1991.

[Zha94] G. Zhang. Tight worst-case performance bound for AFB_k. Technical Report 015, Institute of Applied Mathematics, Academia Sinica, Beijing, china, May 1994.

Index