From Computers and Intractability
by Garey & Johnson

guarantee for a problem. Finally, in Section 6.3, we discuss some practical considerations concerning the applicability of this type of result, and we mention some recent theoretical work directed toward analyzing the "average performance" of heuristic algorithms.

6.1 Performance Guarantees for Approximation Algorithms

Let us begin by presenting a formal description of what we will mean by an "optimization problem."

A *combinatorial optimization problem* Π is either a *minimization problem* or a *maximization problem* and consists of the following three parts:

(1) a set D_Π of *instances*;

(2) for each instance $I \in D_\Pi$, a finite set $S_\Pi(I)$ of *candidate solutions* for I; and

(3) a function m_Π that assigns to each instance $I \in D_\Pi$ and each candidate solution $\sigma \in S_\Pi(I)$ a positive rational number $m_\Pi(I,\sigma)$, called the *solution value* for σ.

If Π is a minimization [maximization] problem, then an *optimal solution* for an instance $I \in D_\Pi$ is a candidate solution $\sigma^* \in S_\Pi(I)$ such that, for all $\sigma \in S_\Pi(I)$, $m_\Pi(I,\sigma^*) \leqslant m_\Pi(I,\sigma)$ $[m_\Pi(I,\sigma^*) \geqslant m_\Pi(I,\sigma)]$. We will use $\mathrm{OPT}_\Pi(I)$ to denote the value $m_\Pi(I,\sigma^*)$ of an optimal solution for I (usually dropping the subscript Π when the problem is clear from context).

An algorithm A is an *approximation algorithm* for Π if, given any instance $I \in D_\Pi$, it finds a candidate solution $\sigma \in S_\Pi(I)$. The value $m_\Pi(I,\sigma)$ of the candidate solution σ found by A when applied to I will be denoted by $A(I)$. If $A(I) = \mathrm{OPT}(I)$ for all $I \in D_\Pi$, then A is called an *optimization algorithm* for Π.

These definitions can be illustrated by considering our old friend the traveling salesman problem. It is a minimization problem, and the set of instances consists of all finite sets of cities together with their intercity distances. The candidate solutions for a particular instance are all the permutations of the given cities. The solution value for such a permutation is the length of the corresponding tour. Thus an approximation algorithm for this problem need only find some permutation of the given set of cities, whereas an optimization algorithm must always find a permutation that corresponds to a minimum length tour.

If, as in this case, the optimization problem is NP-hard, then we know that a polynomial time optimization algorithm cannot be found unless P = NP. A more reasonable goal is that of finding an approximation algorithm A that runs in low-order polynomial time and that has the property that, for all instances I, $A(I)$ is "close" to $\mathrm{OPT}(I)$. The following example illustrates the type of results we will be interested in.

Consider the "bin packing" problem: Given a finite set $U = \{u_1, u_2, \ldots, u_n\}$ of "items" and a rational "size" $s(u) \in [0,1]$ for each item $u \in U$, find a partition of U into disjoint subsets $U_1, U_2, \ldots, U_k$ such that the sum of the sizes of the items in each U_i is no more than 1 and such that k is as small as possible. We can view each subset U_i as specifying a set of items to be placed in a single unit-capacity "bin," with our objective being to pack the items from U in as few such bins as possible.

This problem is NP-hard in the strong sense (it contains 3-PARTITION as a special case), so there is little hope of finding even a pseudo-polynomial time optimization algorithm for it. However, there are a number of simple approximation algorithms for it that are worth considering.

One of these is known as the "First Fit" algorithm. Imagine that we start with an infinite sequence $B_1, B_2, \ldots$ of unit-capacity bins, all of which are empty. The algorithm then places the items into the bins, one at a time, in order of increasing index. It does so according to the following simple rule: always place the next item u_i into the lowest-indexed bin for which the sum of the sizes of the items already in that bin does not exceed $1 - s(u_i)$. In other words, u_i is always placed into the first bin in which it will fit (without exceeding the bin capacity). Figure 6.1 shows an example, where each item is represented by a rectangle having height proportional to its size.

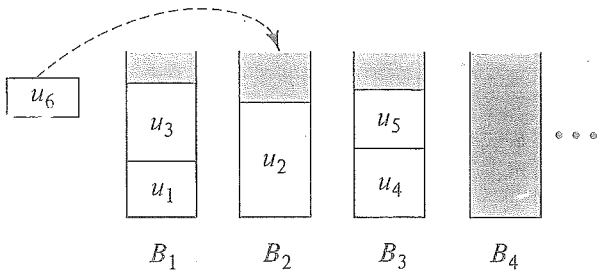

Figure 6.1 An example of a First Fit placement, where u_6 is placed in bin B_2 since that is the lowest indexed bin in which it fits.

Intuitively this seems to be a very natural and reasonable algorithm. It never starts a new bin until all the nonempty bins are too full. What can be proved about its performance?

A first observation relates the number of bins used by First Fit to a natural function of the problem parameters. Let us use "FF" as an abbreviation for First Fit. Then we have that, so long as $FF(I) > 1$,

This is because ther
packing whose conte
higher indexed such
could not have beer
essentially the best p
form $U = \{u_1, u_2, \ldots$
items will fit in the
item sizes is $(n/2) +$
choosing $\epsilon > 0$ suitabl

This observation
can be relative to an

We thus conclude th

However, First Fit a
following theorem fi
1974]:

Theorem 6.1 For all

Furthermore, there e

Thus Theorem
of the First Fit a
significantly more th
bad. (A slight impr
placing (17/10)OPT
Graham, Johnson, a
of Theorem 6.1, we
given by the theore
Figure 6.2, for wh
bound in the theore

$$\mathrm{FF}(I) < \lceil 2 \sum_{i=1}^{n} s(u_i) \rceil$$

This is because there can be at most one nonempty bin in the First Fit packing whose contents total ½ or less. (If not, the first item to go in the higher indexed such bin would have fit in the lower indexed such bin and could not have been placed elsewhere by First Fit.) That this bound is essentially the best possible is apparent when we consider instances of the form $U = \{u_1, u_2, \ldots, u_n\}$ where $s(u_i) = \frac{1}{2} + \epsilon, 1 \leqslant i \leqslant n$. Here no two items will fit in the same bin, so $\mathrm{FF}(I) = n$, even though the sum of the item sizes is $(n/2) + n\epsilon$, which can be made as close to $n/2$ as desired by choosing $\epsilon > 0$ suitably small.

This observation also gives us a bound on how bad a First Fit packing can be relative to an optimal packing, since we clearly have

$$\mathrm{OPT}(I) \geqslant \lceil \sum_{i=1}^{n} s(u_i) \rceil$$

We thus conclude that, for all instances I,

$$\mathrm{FF}(I) < 2 \cdot \mathrm{OPT}(I)$$

However, First Fit actually obeys a better bound of this form, given by the following theorem from [Johnson, Demers, Ullman, Garey, and Graham, 1974]:

Theorem 6.1 For all instances I of the bin packing problem,

$$\mathrm{FF}(I) \leqslant \frac{17}{10}\mathrm{OPT}(I) + 2$$

Furthermore, there exist instances I with $\mathrm{OPT}(I)$ arbitrarily large such that

$$\mathrm{FF}(I) \geqslant \frac{17}{10}(\mathrm{OPT}(I) - 1)$$

Thus Theorem 6.1 characterizes the asymptotic worst-case performance of the First Fit algorithm. First Fit never differs from optimal by significantly more than 70 percent and it can on occasion be essentially this bad. (A slight improvement on the constant term in the upper bound, replacing $(17/10)\mathrm{OPT}(I) + 2$ by $\lceil (17/10)\mathrm{OPT}(I) \rceil$, is obtained in [Garey, Graham, Johnson, and Yao, 1976]). Although we omit the lengthy proof of Theorem 6.1, we note that worst-case behavior almost as bad as that given by the theorem can be seen from the class of examples described in Figure 6.2, for which $\mathrm{FF}(I) = (5/3)\mathrm{OPT}(I)$. (The proof of the lower bound in the theorem is a complicated extension of these examples.)

Instance I: $U=\{u_1,u_2,\ldots,u_{18m}\}$, $s(u_i)=\begin{cases}1/7+\epsilon & 1\leqslant i\leqslant 6m\\ 1/3+\epsilon & 6m<i\leqslant 12m\\ 1/2+\epsilon & 12m<i\leqslant 18m\end{cases}$

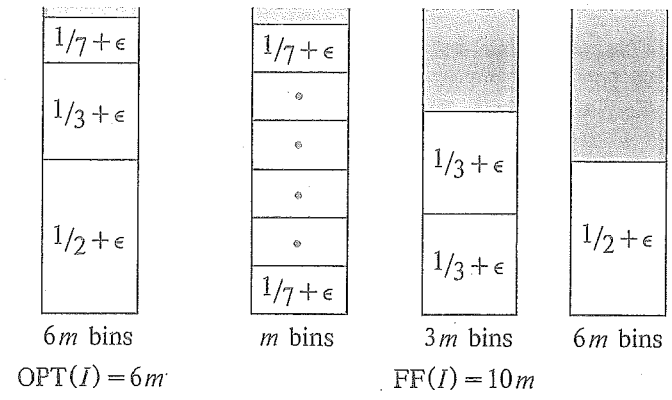

Figure 6.2 Instances I with $\mathrm{OPT}(I)$ arbitrarily large such that $\mathrm{FF}(I)$ equals $(5/3)\mathrm{OPT}(I)$.

These results for First Fit provide a starting point for analyzing approximation algorithms for bin packing. One can now go on to analyze other algorithms that might have better guarantees. An obvious modification to the First Fit algorithm, for example, is that obtained by using the following more sophisticated placement rule: Always place the next item u_i in that bin which has current contents closest to, but not exceeding, $1-s(u_i)$ (choosing the one with lowest index in case of ties). This is known as the "Best Fit" algorithm. Unfortunately, and perhaps surprisingly, Best Fit has essentially the same worst case performance as First Fit [Johnson et al., 1974].

A better approximation algorithm is obtained by observing that the worst performance for First Fit (and Best Fit) seems to occur when the smaller items appear before the larger items in the ordering used by the algorithm. Suppose that, instead of merely taking the items from U in the given order, we first sort them by size and reindex them so that $s(u_1)\geqslant s(u_2)\geqslant\cdots\geqslant s(u_n)$. The algorithm that applies First Fit to such a reordered list is called the "First Fit Decreasing" algorithm (FFD). Its performance is characterized by the following theorem, due to Johnson [1973] (the proof is sketched in [Johnson et al., 1974]):

Theorem 6.2 For all instances I of the bin packing problem,

$$\mathrm{FFD}(I)=\frac{11}{9}\mathrm{OPT}(I)+4$$

Furthermore, there exist instances I with $\mathrm{OPT}(I)$ arbitrarily large such that

$$\mathrm{FFD}(I) = \frac{11}{9}\mathrm{OPT}(I)$$

Thus First Fit Decreasing is guaranteed never to be more than about 22 percent worse than optimal, and it can on occasion be this bad. An identical result holds for the analogous "Best Fit Decreasing" algorithm. Figure 6.3 illustrates a class of examples that suffice to prove the lower bound in both cases.

Instance I: $U = \{u_1, u_2, \ldots, u_{30m}\}$, $s(u_i) = \begin{cases} 1/2+\epsilon & 1 \leqslant i \leqslant 6m \\ 1/4+2\epsilon & 6m < i \leqslant 12m \\ 1/4+\epsilon & 12m < i \leqslant 18m \\ 1/4-2\epsilon & 18m < i \leqslant 30m \end{cases}$

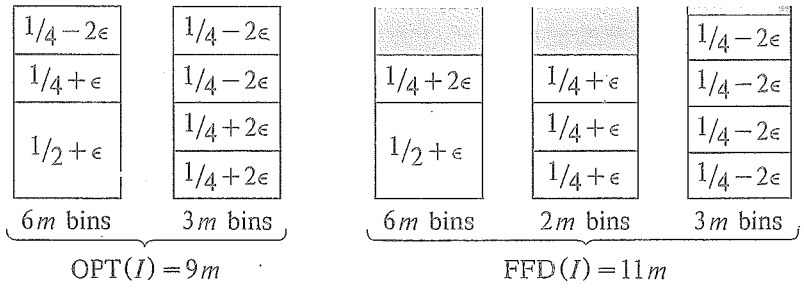

Figure 6.3 Instances I with $\mathrm{OPT}(I)$ arbitrarily large such that $\mathrm{FFD}(I)$ equals $(11/9)\mathrm{OPT}(I)$.

The proof of the upper bound involves an extremely detailed case analysis, whose recapitulation here would require more pages than we have allotted to this entire chapter. (Although such lengthy proofs appear to be the rule for problems similar to bin packing, we note that results like these for other problems have been obtained without such Herculean effort, and indeed even for bin packing much shorter proofs are obtainable if we are willing to settle for weaker bounds.)

Further modifications of First Fit Decreasing have been suggested ([Johnson, 1973], [Yao, 1978a]) in hopes of obtaining a polynomial time approximation algorithm for bin packing with an even better performance guarantee, but no substantially better bound has yet been proved for any of them.

In summary, our analysis of approximation algorithms for the bin packing problem might be described as follows: We started with a straightforward but apparently sensible algorithm and analyzed its performance, both by proving bounds on what could happen in the worst case and by devising examples to verify that these bounds could not be improved. With this

analysis in mind, and especially the insight it provided as to the drawbacks of our initial algorithm, we could then seek alternative algorithms (perhaps just more complicated versions of the original one) and analyze them. We also settled on a general form for our guarantees, in terms of ratios, which was useful for comparison purposes and which seems to express nearness to optimality in a reasonable way. This general approach can serve as a model for our study of other NP-hard optimization problems and indeed has been widely applied (although, of course, on occasion other types of guarantees may be more appropriate or easier to prove, for example, see [Cornuejols, Fisher, and Nemhauser, 1977], [Nemhauser, Wolsey, and Fisher, 1978]).

To formalize this approach, let us make a few more definitions. If Π is a minimization [maximization] problem, and I is any instance in D_Π, we define the ratio $R_A(I)$ by

$$R_A(I) = \frac{A(I)}{\mathrm{OPT}(I)} \qquad \left[R_A(I) = \frac{\mathrm{OPT}(I)}{A(I)} \right]$$

The *absolute performance ratio* R_A for an approximation algorithm A for Π is given by

$$R_A = \inf \{ r \geqslant 1 : R_A(I) \leqslant r \text{ for all instances } I \in D_\Pi \}$$

The *asymptotic performance ratio* R_A^∞ for A is given by

$$R_A^\infty = \inf \left\{ r \geqslant 1 : \begin{array}{l} \text{for some } N \in \mathbb{Z}^+,\ R_A(I) \leqslant r \text{ for all} \\ I \in D_\Pi \text{ satisfying } \mathrm{OPT}(I) \geqslant N \end{array} \right\}$$

Notice that we have defined these ratios in such a way that the ratio for a minimization problem is the reciprocal of that for a maximization problem. This has been done so that we will have a uniform scale on which to consider approximation algorithms for different types of problems, always having $1 \leqslant R_A \leqslant \infty$ and $1 \leqslant R_A^\infty \leqslant \infty$, with a ratio that is closer to 1 indicating better performance.

Notice also that R_A need not equal R_A^∞. Although Theorem 6.2 shows that $R_{\mathrm{FFD}}^\infty = 11/9$, it is easy to give instances I for which $\mathrm{OPT}(I) = 2$ and $\mathrm{FFD}(I) = 3$, so that $R_{\mathrm{FFD}} \geqslant 3/2$. The asymptotic ratios seem to be the more important ones for bin packing, although for other problems the absolute ratios may be more appropriate, or it may be the case that $R_A = R_A^\infty$ for all the approximation algorithms in which we are interested. At any rate, it will be convenient to have both types of ratios available, and the differences between them are worth keeping in mind when analyzing an approximation algorithm.

As a second example, let us return once more to the traveling salesman problem, only this time with an added restriction. An instance I is still a set C of cities and a specification of the distances between them, but we also require that these distances obey the "triangle inequality"; i.e., for every triple a, b, c of cities from C,