An Efficient Approximation Scheme For The One-Dimensional Bin-Packing Problem

*Narendra Karmarkar and Richard M. Karp**

University of California
Berkeley, CA 94720

1. Abstract

We present several polynomial-time approximation algorithms for the one-dimensional bin-packing problem, using a subroutine to solve a certain linear programming relaxation of the problem. Our main results are as follows:

There is a polynomial-time algorithm A such that $A(I) \leq OPT(I) + O(log^2 OPT(I))$. There is a polynomial-time algorithm A such that, if $m(I)$ denotes the number of distinct sizes of pieces occurring in instance I, then $A(I) \leq OPT(I) + O(log^2 m(I))$. There is an approximation scheme which accepts as input an instance I and a positive real number ε, and produces as output a packing using as most $(1+\varepsilon)\,OPT(I) + O(\varepsilon^{-2})$ bins. Its execution time is $O(\varepsilon^{-c}\, n \log n)$, where c is a constant. These are the best asymptotic performance bounds that have been achieved to date for polynomial-time bin-packing. Each of our algorithms makes at most $O(log\ n)$ calls on the LP relaxation subroutine and takes at most $O(n \log n)$ time for other operations. The LP relaxation of bin packing was solved efficiently in practice by Gilmore and Gomory. We prove its membership in P, despite the fact that it has an astronomically large number of variables.

2. Introduction

An instance I of the one-dimensional bin packing problem is specified by n pieces, each of which has a size which is a rational number between 0 and 1. The object is to pack the pieces into a minimum number of bins, subject to the constraint that the sum of the sizes of the pieces in each bin is less than or equal to 1. Let $OPT(I)$ denote the minimum number of bins required in a solution to instance I. An approximation scheme A for the bin packing problem accepts as input an instance I and produces as output a packing which solves instance I. Let $A(I)$ denote the number of bins required by this packing.

Most of the past work on the one-dimensional bin-packing problem has been concerned with approximation algorithms which operate in time $O(n \log n)$ and produce approximations satisfying an inequality of the form $A(I) \leq C\ OPT(I) + o(OPT(I))$. In a series of papers published appearing during the period 1973-1980, the best achievable constant C was gradually reduced from 1.7 to 1.18333... .

In 1981 Fernandez de la Vega and Lueker [2] published a paper which represented a sharp departure from this line of research. They exhibited an approximation scheme which accepts as input an instance I and a positive real number ε, and produces a packing of cost $\leq (1+\varepsilon)\,OPT(I) + O(\varepsilon^{-2})$. The execution time of their method is linear in n, the number of pieces, but grows worse than exponentially as a function of $\frac{1}{\varepsilon}$. Johnson [5] then pointed out that, by letting ε depend on the instance I, one could use any such scheme to construct a polynomial-time algorithm A such that $A(I) \leq OPT(I) + o(OPT(I))$ and, using a particular scheme suggested by the present writers, he achieved $A(I) \leq OPT(I) + O(OPT(I)^{1-\delta})$, where δ is a certain positive constant.

The present paper achieves further progress in the directions initiated by Fernandez de la Vega and Lueker [2] and Johnson [4]. Our results depend upon the ability to solve a certain linear programming relaxation of the bin-packing problem in polynomial time. The solutions of this linear program represent packings in which a given bin configuration is allowed to be used a fractional, rather than integral, number of times. This linear program was introduced by Eisemann [1]. Gilmore and Gomory [3] demonstrated that it can be solved efficiently in practice by a "column generation" method, and used this method successfully for the approximate solution of bin-packing problems arising in the paper industry. To solve this linear program in polynomial time, despite the fact that it has a vast number of variables, we use a variant of the ellipsoid method due to Grotschel, Lovasz and Schrijver [4]. Our algorithms also employ three techniques introduced by Fernandez de la Vega and Lueker (elimination of small pieces, linear grouping and rounding of fractional solutions), as well as two new techniques (geometric grouping and extraction of residual problems) which are essential for our sharpest results.

The execution times of our algorithms are stated in terms of a polynomial-bounded function $T(m, n)$ derived from the time bound for our fractional bin-packing algorithm. In addition, $n(I)$ denotes the number of pieces occurring in instance I, $m(I)$ denotes the number of distinct sizes, $\alpha(I)$ denotes the size of the smallest piece and $SIZE(I)$ denotes the sum of the sizes of all the pieces. The main results are as follows:

*Research supported by NSF Grant MCS81-05217

(1) There is an approximation algorithm A which runs in time $O(T(SIZE(I), n(I))$ such that, for all I, $A(I) \le OPT(I) + O(log^2\ OPT(I))$.

(2) There is an approximation algorithm A which runs in time $O(T(m(I), n(I))$ such that, for all I, $A(I) \le OPT(I) + O(log^2\ m(I))$.

(3) For $0 < \alpha < 1$ there is an approximation algorithm A which runs in time $O(T(SIZE(I)^{(1-\alpha)}, n(I))$ such that, for all I, $A(I) \le OPT(I) + O(OPT(I)^{\alpha})$.

(4) For every $\varepsilon > 0$ there is an approximation algorithm A which runs in time $O(T(\varepsilon^{-2}, n(I))$ such that, for all I, $A(I) \le (1 + \varepsilon)\ OPT(I) + O(\varepsilon^{-2})$.

Unfortunately, the best bound we can currently give for $T(m, n)$ is

$$T(m, n) = O(m^8 \log m \log^2 n + m^4 n \log m \log n).$$

However, it is entirely possible that a blend of our techniques with the column generation method of Gilmore and Gomory will lead to an algorithm which is very efficient in practice.

3. Fractional Bin Packing

Consider an instance I of the bin-packing problem in which there are n pieces of m different types. All pieces of a given type are of the same size. For $i = 1, 2, \ldots, m$ let b_i be the number of pieces of type i, and let s_i be the common size of these pieces. Define a *configuration* as a multiset of piece types capable of being packed within a bin. Let the number of distinct configurations be q, and let a_{ij} denote the number of pieces of type i in configuration j. The fractional bin-packing problem is the linear program (I) minimize $1 \cdot x$ subject to $x \ge 0$ and $A x \ge b$, where 1 is the vector of all 1's and A is the $m \times q$ matrix (a_{ij}).

Example Suppose $m = 2$, $b_1 = 31$, $b_2 = 7$, $s_1 = 0.3$ and $s_2 = 0.4$. Then $q = 7$, the matrix A is $\begin{bmatrix} 0 & 0 & 1 & 2 & 1 & 2 & 3 \\ 1 & 2 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$, and $b = \begin{bmatrix} 31 \\ 7 \end{bmatrix}$. An optimal solution to the linear program is $x = (0, 0, 0, 7, 0, 0, \frac{17}{3})$.

Let $LIN(I)$ denote the optimal value of the fractional bin-packing problem associated with instance I. We shall study the relationship between $LIN(I)$ and $OPT(I)$. Recall that $SIZE(I) = \sum_{i=1}^{m} s_i b_i$ is the sum of the sizes of the pieces associated with instance I.

Lemma 1 $OPT(I) \le 2\ SIZE(I) + 1$.

Proof For any instance I, there exists a packing in which at most one bin is less than half-full. The cost of the packing is at most $2\ SIZE(I) + 1$.

Lemma 2

$$SIZE(I) \le LIN(I) \le OPT(I) \le LIN(I) + \frac{m(I) + 1}{2}$$

Proof $SIZE(I) \le LIN(I)$. Let x be an optimal solution to the linear program. Let s be the m-vector whose ith component is s_i. Then $LIN(I) = 1 \cdot x \ge (s^T A)\, x \ge s^T b = SIZE(I)$.

$(LIN(I) \le OPT(I))$ Corresponding to an optimal solution of the bin-packing problem there is a feasible solution to the linear program in which x_j denotes the number of occurrences of bins with configuration j. The cost of this solution is $OPT(I)$.

$(OPT(I) \le LIN(I) + \frac{m(I) + 1}{2})$ Recall that, in a basic (extreme point) feasible solution to any linear program, the number of nonzero variables is at most the number of constraints, not counting the nonnegativity constraints on individual variables. Let x be an optimal basic feasible solution to the fractional bin-packing problem for instance I. Then x has at most $m(I)$ nonzero components. We construct from x a packing consisting of a *principal part* and a *residual part*. For each nonzero variable x_j. The principal part of the packing includes $\lfloor x_j \rfloor$ bins with configuration j. The remaining pieces comprise an instance I'. Let $f_j = x_j - \lfloor x_j \rfloor$. Then $SIZE(I') \le LIN(I') = \sum_j f_j$, and it suffices to exhibit a packing for instance I' of cost $\le SIZE(I) + \frac{m(I) + 1}{2}$. A packing of cost $\le m(I)$ can be constructed by starting with one bin of configuration j for each nonzero f_j, and then deleting excess pieces. By Lemma 1, there exists a packing of cost $\le 2\ SIZE(I') + 1$. The better of these two packings has cost

$$\min(m(I), 2\ SIZE(I') + 1)$$
$$= SIZE(I') + \min(m(I) - SIZE(I'), SIZE(I') + 1)$$
$$\le SIZE(I') + \frac{m(I) + 1}{2}.$$

The proof of Lemma 2 suggests a general rounding method for obtaining a packing from a basic feasible solution to the fractional bin-packing problem.

Corollary 1 There is an algorithm which has as input an instance I of the bin-packing problem and a basic feasible solution x to the associated fractional bin-packing problem, and produces as output a packing of cost $\le 1 \cdot x + \frac{m(I) + 1}{2}$. The execution time of this algorithm is $O(n(I) \log n(I))$.

All of our approximation algorithms are based on a polynomial-time algorithm for the approximate solution of the linear program (I).

Theorem 1 There exists a polynomial-time algorithm which accepts as input an instance I of the bin-packing problem together with a positive tolerance h, and produces as output a basic feasible solution of costs $\leq LIN(I) + h$ for the linear program (I). Using a model of random-access computation in which the operations of addition, subtraction and comparison take one unit of time, and the square root, multiplication and division operations each take time proportional to the length of the (longer) operand, the execution time of the algorithm is

$$O(m^8 \log m \log^2 (\frac{m\ n}{ah}) + \frac{m^4\ n \log m}{h} \log (\frac{m\ n}{ah})).$$

A randomized version of the algorithm runs in expected time

$$O(m^7 \log m \log^2 (\frac{m\ n}{ah}) + \frac{m^4\ n \log m}{h} \log (\frac{m\ n}{ah})).$$

The proof of Theorem 1 is deferred to the final section of the paper.

4. Elimination and Grouping

In this section we present three basic techniques for the construction of bin-packing algorithms. The first two of these are due to Fernandez de la Vega and Lueker [2].

Elimination of Small Pieces

Each of our bin-packing algorithms proceeds by putting aside all sufficiently small pieces, packing the remaining pieces and, finally, inserting the small ones. The following lemma is used to justify this tactic.

Lemma 3 Let I be an instance of the bin-packing problem, and let g be a real number between 0 and 1. Designate each piece as *large* if its size is $> \frac{g}{2}$ and *small* otherwise. Consider a process which starts with a given packing of the large pieces into bins, and then inserts the small pieces, starting a new bin only when necessary. If the cost of the given packing of the large pieces is A, then the cost of the packing resulting from the process is $\leq max(A, (1+g)\ OPT(I) + 1)$.

Proof Let B be the cost of the packing. If the insertion of the small pieces does not require the use of a new bin then $B = A$. If a new bin is required, then, in the resulting packing, there is at most one bin in which the sum of the piece sizes is less than $1 - \frac{g}{2}$. It follows that $SIZE(I) \geq (1 - \frac{g}{2})(B - 1)$, so that

$$B \leq \frac{1}{1-\frac{g}{2}} SIZE(I) + 1 \leq (1+g)\ OPT(I) + 1$$

Linear Grouping

Define a partial order on bin-packing instances as follows:

$I \leq J$ if there exists a one-to-one function $f(x)$ from I into J such that $SIZE(x) \leq SIZE(f(x))$, for each item $x \in I$

clearly,

$$I \leq J \Rightarrow \begin{cases} OPT(I) \leq OPT(J) \\ LIN(I) \leq LIN(J) \\ SIZE(I) \leq SIZE(J) \end{cases}$$

The following process is called linear grouping with parameter k. Its purpose is to decrease the number of different types of pieces, without increasing the piece sizes too much.

Let I be an instance of the bin-packing problem and let k be positive integer. Divide the set I into groups $G_1, G_2, \ldots, G_q$ so that G_1 contains the k largest pieces, G_2 contains the next k largest pieces and so on.

Hence $G_1 \geq G_2 \geq \cdots \geq G_q$

and $|G_i| = k$ for all except the last group.

Let G_i' be the multi-set of items obtained by increasing the size of each piece in group G_i to the maximum size of a piece in that group.

Hence $G_1 \geq G_2' \geq G_2 \geq \cdots \geq G_q' \geq G_q$

$\therefore$ $\bigcup_{i=2}^{q} G_i' \leq I \leq \bigcup_{i=1}^{q} G_i'$

Let $J = \bigcup_{i=2}^{q} G_i'$ and $J' = G_1$

$\therefore$ $J \leq I \leq J \cup J'$

$\therefore$ $OPT(J) \leq OPT(I) \leq OPT(J \cup J')$
$\leq OPT(J) + OPT(J') \leq OPT(J) + k$

Thus we have proved the following lemma.

Lemma 4 $OPT(J) \leq OPT(I) \leq OPT(J) + k$
$LIN(J) \leq LIN(I) \leq LIN(J) + k$
$SIZE(J) \leq SIZE(I) \leq SIZE(J) + k$

Geometric Grouping

The following refinement of linear grouping is called geometric grouping with parameter k. It is one of the main innovations of the present paper.

Let I be an instance of the bin-packing problem. Recall that $a(I)$ denotes the size of the smallest piece. For $r = 0, 1, \cdots \lfloor \log_2 \frac{1}{a(I)} \rfloor$ let I_r be the instance consisting of those pieces from I whose sizes lie in the interval $(2^{-(r+1)}, 2^{-r})$. Let J_r and J_r' be the instances obtained by applying linear grouping with parameter $k \cdot 2^r$ to I_r.

Let $J = \bigcup_r J_r$ and $J' = \bigcup_r J'_r$

Lemma 5 $OPT(J) \leq OPT(I) \leq OPT(J) + k \lfloor \log_2 \frac{1}{a(I)} \rfloor$
$LIN(J) \leq LIN(I) \leq LIN(J) + k \lfloor \log_2 \frac{1}{a(I)} \rfloor$
$SIZE(J) \leq SIZE(I) \leq SIZE(J) + k \lfloor \log_2 \frac{1}{a(I)} \rfloor$
and $m(J)$ (number of distinct item-sizes in J)
$\leq \frac{2}{k} SIZE(I) + \lfloor \log_2 \frac{1}{a(I)} \rfloor$

Proof From Lemma 4,

$J_r \le I_r \le J_r \cup J_r'$

$\therefore \quad J \le I \le J \cup J'$

$\therefore \quad OPT(J) \le OPT(I) \le OPT(J \cup J')$
$\le OPT(J) + OPT(J')$

$OPT(J') \le \sum_r OPT(J_r')$

Each non-empty J_r' contains $k \cdot 2^r$ pieces each of size less than 2^{-r}. Hence J_r' can be packed into at most k bins.

$\therefore \quad OPT(J') \le k \cdot \lceil \log_2 \frac{1}{a(I)} \rceil$

$\therefore \quad OPT(J) \le OPT(I) \le OPT(J) + k \cdot \lceil \log_2 \frac{1}{a(I)} \rceil$

The proof of other two inequalities is similar.

For each r

$$SIZE(I_r) \ge 2^{-(r+1)} \cdot n(I_r) \ge 2^{-(r+1)}[(m(J_r)-1) \cdot k \cdot 2^r]$$

$\therefore \quad m(J_r) - 1 \le \frac{2}{k} SIZE(I_r)$

$\therefore \quad m(J) \le \frac{2}{k} SIZE(I) + \lceil \log_2 \frac{1}{a(I)} \rceil$

Now we present another variation of Geometric Grouping:

Sort the set I of items in non-increasing order of piece-sizes. Starting from the largest piece, put as may items in group G_1 as necessary to make the size of G_1 equal or exceed k. Repeating the same operation on the remaining pieces, create a sequence of groups $G_2, G_3, \cdots G_q$. Let $l_i = |G_i|$. Then $k + 1 \le l_1 \le l_2 \le \cdots l_q \le \frac{k}{a(I)}$. Now we can no longer claim that $G_i \ge G_{i+1}$, since G_{i+1} may contain more items than G_i. To remedy this problem, we partition G_i, $i = 2, \cdots q$, as follows:

Let ΔG_i = the set of smallest $(l_i - l_{i-1})$ pieces in G_i.

$\therefore \quad |G_i - \Delta G_i| = l_{i-1} = |G_{i-1}|$

Let G_i' be the multi-set obtained by increasing the size of each piece in $G_i - \Delta G_i$ to the maximum size of a piece in $G_i - \Delta G_i$.

$$\therefore \quad \left.\begin{array}{l} G_i' \le G_{i-1} \\ G_i \le G_i' \cup \Delta G_i \end{array}\right\} \; i = 2, 3, \cdots q.$$

Define $J = \bigcup_{i=2}^{q} G_i$, $J' = \bigcup_{i=2}^{q} \Delta G_i' \cup G_1$

$\therefore \quad J \le I \le J \cup J'$

$\therefore \quad SIZE(J) \le SIZE(I) \le SIZE(J \cup J')$
$\le SIZE(J) + SIZE(J')$

Now we establish a bound on $SIZE(J')$.

By definition of G_i, the total size of the largest $l_i - 1$ pieces in G_i is less than k, and ΔG_i is the set of the smallest $l_i - l_{i-1}$ pieces in G_i

$\therefore \quad SIZE(\Delta G_i) \le \frac{l_i - l_{i-1}}{l_i - 1} \cdot k$

$\therefore \quad SIZE(J') = SIZE(G_1) + \sum_{i=2}^{q} SIZE(\Delta G_i)$

$\le k[\frac{l_1}{l_1 - 1} + \sum_{i=2}^{q} \frac{l_i - l_{i-1}}{l_i - 1}]$

Note that $\frac{l_i - l_{i-1}}{l_i - 1} \le \frac{1}{l_{i-1}} + \frac{1}{l_{i-1}+1} + \cdots \frac{1}{l_i - 1}$

$\therefore \quad SIZE(J') \le k[1 + \frac{1}{l_1 - 1} + \frac{1}{l_1} + \cdots \frac{1}{l_q - 1}]$

$\le k[1 + \ln \frac{l_q - 1}{l_1 - 2}] \le k[1 + \ln \frac{\frac{k}{a(I)} - 1}{k - 1}]$

$\le k[1 + \ln \frac{1}{a(I)}] + 1$

Also,

$LIN(J') \le OPT(J') \le 2.SIZE(J') + 1 \le 2k[2 + \ln \frac{1}{a(I)}]$

$SIZE(J) \ge \sum_{i=2}^{q} k \cdot \frac{l_{i-1}}{l_i}$

$\ge k \cdot \sum_{i=2}^{q} [1 - \frac{l_i - l_{i-1}}{l_i}]$

$\ge k(q - 1) - k[\frac{1}{l_1 + 1} + \frac{1}{l_1 + 2} \cdots + \frac{1}{l_q}]$

$\ge k(q - 1) - k \ln \frac{l_q}{l_1} \ge k[(q - 1) - \ln \frac{1}{a(I)}]$

$\therefore \quad m(J) = q - 1 \le \frac{SIZE(J)}{k} + \ln \frac{1}{a(I)}$

Thus we have proved the following theorem:

Theorem 2

1. $SIZE(J) \le SIZE(I) \le SIZE(J) + k[2 + \ln \frac{1}{a(I)}]$
2. $LIN(J) \le LIN(I) \le LIN(J) + 2k[2 + \ln \frac{1}{a(I)}]$
3. $OPT(J) \le OPT(I) \le OPT(J) + 2k[2 + \ln \frac{1}{a(I)}]$
4. $m(J) \le \frac{SIZE(J)}{k} + \ln \frac{1}{a(I)}$

5. The Algorithms

The following algorithms invoke a Fractional Bin-Packing procedure which accepts as input an instance I of the bin-packing problem and a positive tolerance h, and produces as output a basic feasible solution of cost less than or equal to $LIN(I) + h$ for the linear program (I). In stating time bounds we postulate that, whenever this subroutine is executed with $n(I) \le n$, $m(I) \le m$, $a(I) \ge \frac{1}{n}$ and $h \ge 1$, its execution time does not exceed $T(m, n)$, where T is a fixed monotonic increasing polynomial-bounded function. A particular implementation of this subroutine is given in Section 6, along with a suitable choice of the function T.

Our first bin-packing algorithm accepts as input an instance I and a positive real number ε.

ALGORITHM 1

1. Discard all pieces of size $\le max(\frac{1}{n}, \frac{\varepsilon}{2})$. Call the resulting instance J.
2. Perform linear grouping with $k = \lceil n(J)\varepsilon^2 \rceil$. Call the resulting instances K and K'.
3. Pack each of the k pieces in the instance K' into a bin by itself.

4. Apply the Fractional Bin-Packing subroutine to instance K with tolerance $h = 1$. Let x be the basic feasible solution produced as output.
5. Using the rounding method of Corollary 1, construct from x a packing for instance K using at most $1 \cdot x + \frac{m(K)+1}{2}$ bins.
6. By reducing piece sizes as necessary, create a packing for instance J from the packing for K together with the k bins created in Step 3.
7. Create a packing for instance I by inserting the pieces discarded in Step 1, using a new bin only when necessary.

Theorem..3 The execution time of ALGORITHM 1 is $O(n(I)\log n(I) + T(\frac{1}{\varepsilon^2}, n(I)))$

Let $A(I)$ denote the cost of the packing produced by ALGORITHM 1. Then

$$A(I) \le (1+\varepsilon)\, OPT(I) + \frac{1}{2\varepsilon^2} + 3$$

Proof The proof proceeds via the following inequalities:

(i) $OPT(K) \le OPT(J) \le OPT(I)$.

(ii) $OPT(J) \ge SIZE(J) \ge \varepsilon n(J)$.

(iii) $k = \lceil n(J)\varepsilon^2 \rceil \le \varepsilon\, OPT(J) + 1 \le \varepsilon\, OPT(I) + 1$.

(iv) $m(K) \le \frac{n(K)}{k} \le \frac{n(K)}{\lceil n(J)\varepsilon^2 \rceil} \le \frac{n(J)}{\lceil n(J)\varepsilon^2 \rceil} \le \frac{1}{\varepsilon^2}$.

(v) The execution time of the Fractional Bin-Packing algorithm is $\le T(m(K), n(K)) \le T(\frac{1}{\varepsilon^2}, n(I))$.

(vi) Since time $O(n(I)\log n(I))$ suffices for all the operations except the Fractional Bin-Packing subroutine, the execution time of ALGORITHM 1 is $O(n(I)\log n(I) + T(\frac{1}{\varepsilon^2}, n(I)))$.

(vii) $1 \cdot x \le LIN(K) + 1 \le OPT(K) + 1 \le OPT(I) + 1$.

(viii) The cost of the packing produced at Step 6 is

$$\le 1 \cdot x + \frac{m(K)+1}{2} + k$$

$$\le (OPT(I)+1) + \frac{1+\frac{1}{\varepsilon^2}}{2} + \varepsilon\, OPT(I) + 1.$$

(ix) By Lemma 3, the cost of the packing produced at Step 7 is

$$A(I) \le max((1+\varepsilon)\, OPT(I) + \frac{1}{2\varepsilon^2} + 3, (1+\varepsilon)\, OPT(I) + 1)$$

$$\le (1+\varepsilon)\, OPT(I) + \frac{1}{2\varepsilon^2} + 3.$$

Our second bin-packing algorithm has two parameters: a positive integer k and a positive real number g.

ALGORITHM 2

1. Eliminate all pieces of size $\le g$.
2. While $SIZE(I) > 1 + \frac{1}{1-\frac{1}{k}} \ln \frac{1}{g}$ do

   Perform geometric grouping with parameter k to create instances J and J'. Pack J' in at most $2k[2 + \ln \frac{1}{g}]$ bins.

   Execute the Fractional Bin-Packing subroutine on instance J with tolerance $h = 1$; let x be the resulting basic feasible solution.

   For each k such that $x_j \ge 1$, create $\lfloor x_j \rfloor$ bins with configuration j and delete the pieces so packed.
3. Pack the remaining pieces in at most $2 + \frac{2}{1-\frac{1}{k}} \ln \frac{1}{g}$ bins.
4. Insert the pieces eliminated at Step 1, using a new bin only when necessary.

Analysis of ALGORITHM 2

Let t denote the number of times the body of the while loop is executed. For $i = 1, 2, \cdots t$ define problem instances I_i, J_i, J'_i and non-negatives integers X_i and Y_i associated with the ith execution of the body of the while loop; I_i is the instance occurring at the beginning of this execution of the body, J_i and J'_i are the instances that result from the application of geometric grouping to I_i, X_i is the number of bins created using x, the solution produced by the Fractional Bin-Packing subroutine applied to instance J_i, and Y_i is the number of bins required to pack J'_i. In particular $I_i = I$.

We first derive an upper bound on t, the number of iterations.

$$SIZE(I_{i+1}) \le LIN(I_{i+1}) \le \sum_j x_j - \lfloor x_j \rfloor \le$$

$$m(J_i) \le \frac{SIZE(J_i)}{k} + \ln \frac{1}{g} \le \frac{SIZE(I_i)}{k} + \ln \frac{1}{g}$$

$$\therefore\ SIZE(I_{i+1}) \le (\tfrac{1}{k})^i\, SIZE(J) + \ln \tfrac{1}{g}[1 + \tfrac{1}{k} + \cdots (\tfrac{1}{k})^{i-1}]$$

$$\le (\tfrac{1}{k})^i\, SIZE(I) + \frac{1}{1-\frac{1}{k}} \ln \frac{1}{g}$$

but $SIZE(I_t) > 1 + \frac{1}{1-\frac{1}{k}} \ln \frac{1}{g}$

$$\therefore\ t \le \frac{\ln SIZE(I)}{\ln k} + 1$$

To derive an upper bound on execution time, note that in the i th execution of the body of the while loop, the time to execute the Fractional Bin-Packing subroutine is $\le T(m(J_i), n(J_i))$ and the time for all other operations is $O(n(I)\log n(I))$. $m(I_i)$ decreases according to a geometric series. Thus the execution time of the algorithm is

$$(T(\frac{SIZE(I)}{k} + \ln \frac{1}{g}, n(I)) + n(I)\log n(I)).$$

To bound the error in the number of bins produced by the algorithm, we use the following inequalities:

$$LIN(I_{i+1}) \le LIN(J_i) + 1 - X_i \le LIN(I_i) + 1 - X_i$$

$$\therefore\ \sum_{i=1}^{t} X_i \le LIN(J) + t$$

The number of bins in the packing produced after Step 3 is

$$\leq \sum_{i=1}^{t} X_i + t[2k(2 + \ln \frac{1}{g})] + 2 + \frac{2}{1 - \frac{1}{k}} \ln \frac{1}{g}$$

Hence by Lemma 3,

$$A(I) \leq \max\left\{(1 + 2g)\, OPT(I) + 1,\right.$$
$$\left. OPT(I) + [1 + \frac{\ln SIZE(I)}{\ln k}][1 + 4k + 2k \ln \frac{1}{g}] + 2 + \frac{2}{1 - \frac{1}{k} \ln \frac{1}{g}}\right\}$$

Certain special choices of parameters k and g in ALGORITHM 2 lead to noteworthy results.

If we choose $k = 2$ and $g = \frac{1}{SIZE(I)}$, we get the following theorem:

Theorem 4 There is polynomial time algorithm A such that $A(I) \leq OPT(I) + O(\log^2 OPT(I))$. It runs in time $O(T(\frac{n(I)}{2}, n(I)) + n(I) \log n(I)))$. In particular, it makes $O(\log n)$ calls on the Fractional Bin-Packing subroutine.

It is possible to obtain a trade-off between time and error by the following choice of parameters:

Let $0 < \alpha < 1$ $k = SIZE(I)^{\alpha}$, $\frac{1}{g} = SIZE(I)^{1-\alpha}$

Then $A(I) \leq OPT(I) + O(OPT(I)^{\alpha} \cdot \log(OPT(I)))$ and the execution time is $O(T(2\, n(I)^{1-\alpha}, n(I)) + n(I) \ln n(I))$.

Finally we point out a variant of ALGORITHM 2 that is advantageous when $m(I)$ is small.

ALGORITHM 3

1. Eliminate all pieces of size $\leq \frac{\log^2 m(I)}{SIZE(I)}$. Call the resulting instance K.
2. If $m(K) \leq SIZE(K)$ then
   Execute the Fractional Bin-Packing subroutine on instance K with $h = 1$; let x be the resulting basic feasible solution.
   For each j such that $x_j \geq 1$, creates $\lfloor x_j \rfloor$ bins with configuration j, and delete the pieces so packed from the set of remaining pieces.
3. Execute Step 2 of ALGORITHM 2 on the instance defined by the set of remaining pieces.
4. Insert the pieces eliminated at Step 1, using a new bin only when necessary.

Theorem 5 The execution time of ALGORITHM 3 is $O(T(m(I), n(I)) + n(I) \log n(I)) \log m(I)$ If $A(I)$ is the cost of the packing produced by ALGORITHM 3 then $A(I) \leq OPT(J) + O(\log^2 m(I))$.

The proof of Theorem 4 is omitted; it is quite similar to the proof of Theorem 3.

6. A Fractional Bin-Packing Algorithm

This section is devoted to the construction of a polynomial-time algorithm which accepts as input an instance I of the bin-packing problem together with a positive tolerance h, and produces as output a basic feasible solution of cost $\leq LIN(I) + h$ for the linear program I. Using a model of random-access computation in which the operations of addition, subtraction and comparison take one unit of time, and the square root, multiplication and division operations each take time proportional to the length of the (longer) operand, the execution time of the algorithm is

$$O(m^8 \ln m \log^2(\frac{m\, n}{a\, h}) + \frac{m^4 n \log m}{h} \log(\frac{m\, n}{a\, h}))$$

where a, m and n are abbreviations for $a(I)$, $m(I)$ and $n(I)$, respectively. A randomized version of the algorithm runs in expected time

$$O(m^7 \log m \log^2(\frac{m\, n}{a\, h})) + \frac{m^4 n \log m}{h} \log(\frac{m\, n}{a\, h}))$$

The GLS method

Our fractional bin-packing algorithm is based on the Grotschel-Lovasz-Schrijver (GLS) version of the ellipsoid method for the solution of linear programming problems. The input to the GLS algorithm consists of: a bounded convex polyhedron $P \leq R^n$ and a profit vector $c \in R^n$; a positive real tolerance t; points z_1 and z_2 in R^n and positive real numbers r and R such that $B(z_1, r) \leq P \leq B(z_2, R)$, where $B(\sigma, \zeta)$ denotes the ball with center σ and radius ζ. The output is point $p \in P$ such that $c \cdot p \geq \max c \cdot y - t$ Thus p is a near-optimal feasible solution to the linear programming problem maximize $c \cdot y$ subject to $y \in P$.

The algorithm requires a "separating hyperplane oracle" which, given a point $z \in R^n$, either determines that $z \in P$ or else produces d such that a $d \cdot z < d \cdot y$ for all $y \in P$.

The algorithm constructs a sequence $E_0, E_1, \ldots,$ of ellipsoids in R^n, with centers $y_0, y_1, \ldots,$ such that the following inductive assertion hold for each k: E_k contains every point $y \in P$ such that

$$c \cdot y > \max\{c \cdot y_j \mid y_j \in P, j = 1, 2, \ldots, k - 1\}$$

Thus, after iteration $k - 1$ the search for an improved feasible solution need only consider points in E_k.

The inital ellipsoid E_0 is the ball $B(z, R)$. At each iteration an ellipsoid E_k with center y_k will be given. The algorithm intersects this ellipsoid with a half-space of the form $y \cdot h_k \geq y_k \cdot h_k$, and E_{k+1} is an ellispoid circumscribed in a particular way (cf. [ ]) around the half-ellipsoid created by the intersection of E_k with the half-space. The construction of E_{k+1} from E_k and h_k requires $O(n^2)$ arithmetic, operations, and the volume of E_{k+1} is smaller than that of E_k by the factor $e^{-\frac{1}{5n}}$.

The vector h_k is determined by submitting the point y_k to the oracle. If the oracle determines that $y_k \notin P$ then $h_k = c$, and the effect of the iteration is to eliminate points where the value of the objective function is less than or equal to the value at y_k. If the oracle determines that $y_k \in P$ and produces a $d \in R^n$ such that $d \cdot y_k < d \cdot y$ for all $y \in P$, then $h = d$, and the effect of the iterations to eliminate infeasible points.

Let $N = 4n^2 \lceil \ln \frac{2R^2 \|C\|}{rt} \rceil$. It is proven in [4] that, if N iterations are carried out, and if each intermediate result is calculated with $5N$ bits of precision after the binary point, then the problem will be solved within a tolerance of t; i.e., the following inequality will hold:

$$\max\{c \cdot y_k \mid y_k \in P\} \geq \max\{c \cdot y \mid y \in P\} - t.$$

The time to solve the linear program within t, excluding the execution time of the separating hyperplane oracle, can now be estimated. On the assumption that the time to extract a square root or perform an arithmetic operation is proportional to the length of the (longer) operand, the time for each iteration is $O(n^2 N)$, and the time for the entire computation is $O(n^2 N^2)$.

The Dual of the Fractional Bin-Packing Problem

Our object is to solve within an additive error h the linear program min $1 \cdot x$ subject to $x \geq 0$, $A x \geq b$, where the columns of A correspond to the possible bin configurations and b_i is the number of pieces of type i, $i = 1, 2, \ldots, m$. Because this linear program has an astronomically large number of variables we consider instead the dual linear program, which has a constraint for every possible bin configuration, but only m variables. The dual program is: maximize $u \cdot b$ subject to $u \geq 0$ and $u^T A \leq 1$.
The duality theorem of linear programming tells us that the primal and dual programs have the same optimal value: min $1 \cdot x = \max u \cdot b$.

The dual program has the following economic interpretation. The dual variable u_i is the price of every piece of type i. The program seeks to maximize the sum of the prices of all the pieces, subject to the constraints that each price must be nonnegative, and the the sum of the prices of all the pieces in any single bin must be ≤ 1.

We shall solve the dual program within an additive tolerance t using a modified form of the GLS algorithm. Consider the task faced by a separating hyperplane oracle which must determine whether a point u is feasible for the dual and, if not, must exhibit a constraint violated at u. We may assume that $u \geq 0$, since otherwise the task is trivial. The nonnegative vector u is feasible if and only if there is no bin configuration of price greater than 1, where u_i is the price of a piece of type i. Let s_i be the size of a piece of type i, and let s be the m-vector whose i th component is s_i. Then u is feasible if and only if th optimal value of the following knapsack problem is ≤ 1: maximize $v \cdot u$ subject to $v \cdot s \leq 1$, $v \geq 0$, v integer.
Here v represents a bin configuration containing v_i pieces of type i. If u is infeasible and v is an optimal solution to the knapsack problem, then the required violated inequality is $u \cdot v \leq 1$.

To avoid solving arbitrary instances of the NP-hard knapsack problem we modify the GLS algorithm as follows. Each time the separating hyperplane oracle is called to determine whether the m-vector u is feasible, we round u to a nearby vector $\hat{u}$ for which the knapsack problem is easy to solve, and then we determine whether $\hat{u}$ is feasible. This test of the "approximate feasibility" of u will permit the ellipsoid method to solve the dual program within a specified tolerance t in polynomial time.

At each iteration of the modified GLS method an ellipsoid $\bar{E}$ with center $\bar{u}$ is given. A new price vector $\hat{u}$ is created by rounding each component of $\bar{u}$ down to a multiple of $\frac{t}{n}$; i.e., $\hat{u}_i = \frac{t}{n} \lfloor \frac{n}{t} \bar{u}_i \rfloor$. The knapsack problem with this price vector and the given piece sizes is then solved by dynamic programming as follows. For $k = 0, 1, \ldots, 1 + \max_i \hat{u}_i$, let $F(k)$ be the minimum total length of a set of pieces whose total price is $k \cdot \frac{t}{n}$. Then $F(0) = 0$ and

$$F(k) = \min_{k - u_i \geq 0} [F(k - \hat{u}_i) + s_i], \; k > 0.$$

The point $\hat{u}$ is feasible if and only if, for all $k > 1$, $F(k) > 1$. The number of additions, subtractions and comparisons to tabulate F, determine whether $\hat{u}$ is feasible and, if not, recover a bin configuration of price > 1, is $O(\frac{nm}{t})$.

Depending on the outcome of the knapsack calculation, two cases arise:

Case 1 The point $\hat{u}$ is infeasible. In this case the dynamic programming calculation determines a m-vector α, corresponding to some configuration, such that $\hat{u} \cdot \alpha > 1$. Then $\bar{u} \cdot \alpha > \hat{u} \cdot \alpha > 1$, so the point $\bar{u}$ is not feasible. In this case a "feasibility cut" is made, imposing the constraint $u \cdot \alpha \geq \bar{u} \cdot \alpha$.

Case 2 The point $\hat{u}$ is feasible. In this case, $\bar{u}$ may or may not be feasible. Nevertheless an "optimality cut" through $\bar{u}$ is made, imposing the constraint $u \cdot b >= \bar{u} \cdot b$. Since $\bar{u}_i \leq \hat{u}_i + \frac{t}{n}$ for each i, $\bar{u} \cdot b \leq \hat{u} \cdot b + t$. Thus no point of $\bar{E}$ is eliminated if its objective function value exceeds the value at the feasible point $\hat{u}$ be more than t.

The bound on the number of iterations for the GLS algorithm to solve a linear program within a tolerance of t applies to our modified algorithm as well. Moreover, $B(u_0, r) \leq K \leq B(u_1, R)$ where K is the feasible region for the (dual) linear program, each component of u_0 is $\frac{a(I)}{2}$, $r = \frac{a(I)}{2}$ each component of u_1 is $\frac{1}{2}$ and $R = \frac{1}{2}\sqrt{m}$. It follows that the modified algorithm can be terminated after M iterations, with the guarantee that the best feasible solution found achieves a price within t of the optimal price, where $M = 4m^2 \lceil \ln \frac{mn}{at} \rceil$.

The execution time of the modified algorithm, taking into account both the knapsack calculations and the steps within the ellipsoid method proper, is $O(Mm(Mm + \frac{n}{t}))$.

The Fractional Bin-Packing Subroutine

The fractional bin-packing program and its dual are

(I) minimize $1 \cdot x$ subject to $x \geq 0$ and $A x \geq b$

(II) maximize $u \cdot b$ subject to $u \geq 0$ and $u^T A \leq 1$.

Let z be the common optimal value of these two programs.

The Fractional Bin-Packing subroutine produces a basic feasible solution for program (I) of cost $\leq z + h$. The subroutine involves a parameter $t > 0$ which will be specified below. The first step in the subroutine is to execute the modified GLS algorithm on program (II) to obtain a solution u^* of value $\geq z - t$. An important byproduct of this calculation is the ability to eliminate from consideration all but a handful of the vast number of possible bin configurations. Recall that each configuration corresponds to a variable in program (I) and a constraint in program (II) Define a realized linear program (II') which has the same variables and objective function as (II), but includes only the following constraints:

> the constraints $0 \leq u_i \leq 1, i = 1, 2, \ldots, m$ and the constraints of the form $u \cdot v \leq 1$ returned by the separating hyperplane

oracle at those iterations when infeasible points $\tilde{u}$ were presented to it. an observer of the execution just completed of the modified GLS algorithm would have no way to determine whether program (II) or program (II') was being solved. both restriction of the search to the original ellipsoid E_0 and the behavior of the separating hyperplane oracle are valid for both programs. it follows that every claim that can be made about program (II) on the basis of this calculation is also valid for program (II'). Let z' be the optimal value of program (II'). Then $u^* \cdot b \leq z \leq z' \leq u^* \cdot b + t$. Here the inequality $u^* \cdot b \leq z$ follows because u^* is a feasible solution of (II), $z \leq z'$ because (II') is a relaxation of (II), and $z' \leq u^* \cdot b + t$ follows from the properties of the modified GLS algorithm. It follows that $z \leq z' \leq z + t$; thus, little error is committed in considering (II') (or its dual (I'), which represents a fractional bin-packing problem in which only a restricted set of configurations is permitted) instead of (II) or its dual (I).

The Fractional Bin-Packing subroutine continues by gradually eliminating constraints from (II') without increasing the optimal value very much. Eventually, exactly m linearly independent constraints will remain. Each of these constraints will either be of the form $-u_i \leq 0$ or of the form $u \cdot \alpha \leq 1$, where α specifies a bin configuration. The linear program determined by these constraints will be of the form maximize $u \cdot b$ subject to $u^T B \leq r$, where r is a vector of 0's and 1's and B is a nonsingular $m \times m$ matrix. The constraint-elimination process will guarantee that the optimal value of this program will be within a tolerance h of z, the optimal value of (II). The dual of this program has, as its unique optimal basic feasible solution, the vector $x = B^{-1} b$. Every component of x will be either a slack variable corresponding to the inequality $-u_i \leq 0$, or a variable associated with a bin configuration, corresponding to an inequality of the form $u \cdot \alpha \leq 1$. Those components of x corresponding to bin configurations will determine the desired near-optimal bin packing.

At a general step of the constraint-elimination process a linear program $(\bar{II})$ will be given; this program has the same variables and objective function as (II'), but only a subset of its constraints; a feasible solution $\bar{u}$ of $(\bar{II})$ will also be given, such that, if $\bar{z}$ is the optimal value of $(\bar{II})$, then $\bar{z} - t \leq \bar{u} \cdot b \leq \bar{z}$. The constraint-elimination procedure selects a subset S of the set of constraints of (II), and constructs an even more relaxed linear program $(\bar{II})$ by deleting these constraints. A test is then made to determine whether the subset S can be dropped. This is done by using the GLS algorithm to find a feasible solution $\tilde{u}$ for $(\bar{II})$ such that $\tilde{u} \cdot b$ is within t of the optimal value of $(\bar{II})$. If $\tilde{u} \cdot b \leq \bar{u} \cdot b + t$ then the test is successful and the subset of constraints S is permanently dropped. Otherwise the subset of constraints S is retained and the constraint-elimination procedure tries another subset. The process terminates when only m constraints remain.

The following observation determines the method of selecting trial subsets of constraints to be dropped. It is a fundamental fact of linear programming that, in any bounded linear program with m variables, there is a set of m constraints which determine the optimal values; i.e., the optimal value would not increase even if all the other constraints were dropped. If T is this critical set of constraints for the program $(\bar{II})$ and S is a subset of constraints disjoint from T, then the test based on the GLS algorithm will be successful, and the subset S will be dropped.

At a general step the constraint-elimination procedure partitions the set of constraints into $m + 1$ subsets of nearly equal size, and tests these subsets successfully until it locates a subset which can be dropped. By the pigeonhole principle, at least one of these subsets is disjoint from T, and therefore qualifies to be dropped. Moreover, if a random partition into subsets is used and the number of constraints is much greater than m, then the expected number of trials until the first success will be bounded by a constant.

Let the number of constraints in program (II') be Q. Then $Q \leq M + 2m$, where $M = 4m^2 \lceil \ln \frac{m\,n}{at} \rceil$ is an upper bound on the number of iterations when the modified GLS algorithm is applied to program (II). An easy calculation shows that the number of successful trails of the constraint-elimination procedure needed in order to reduced the number of constraints to m is $\leq \lceil \frac{Q}{m+1} \rceil + (M + 1) \ln \leq \lceil \frac{Q}{m+1} \rceil + 1$. Since at least one out of every $m + 1$ successive trials succeeds, the total number of executions of the GLS algorithm is $(m + 1)(\lceil \frac{Q}{m+1} \rceil + (m + 1) \ln \lceil \frac{Q}{m+1} \rceil + 1)$. For the randomized version of the constraint-elimination algorithm, the expected number of executions of the GLS algorithms is $O(m + 1)(1 + \ln \lceil \frac{Q}{m+1} \rceil)$. If the tolerance t within the GLS algorithm is

$$\frac{h}{\lceil \frac{Q}{m+1} \rceil + (m + 1) \ln \lceil \frac{Q}{m+1} \rceil + 2}$$

then the cost of the packing obtained will not exceed the cost of an optimal packing by more than h.

We have now specified a polynomial-time algorithm which accepts as input an instance I of the bin-packing problem and a tolerance h, and produces as output a basic feasible solution of cost $\leq LIN(I) + h$ for the linear program ( ). The main steps of the algorithm are as follows.

(1) Set t equal to

$$\frac{h}{\lceil \frac{Q}{m+1} \rceil + (m+1) \ln \lceil \frac{Q}{m+1} \rceil + 2}.$$

(2) Execute the modified GLS algorithm with tolerance t on the program (II) minimize $u \cdot b$ subject to $u \geq 0$, $u^T A \leq 1$.

(3) Extract from the execution of the modified GLS algorithm a relaxed linear program (II') with the same variables and objective function as (II), but with only $Q \leq M + 2m$ constraints.

(4) Use the constraint-elimination procedure to obtain a linear program (II^*) with the same variables and objective function as (II'), but with only m constraints.

(5) Obtain a near-optimal bin packing by solving the dual of (II^*).

The execution time of this algorithm is

$$O(m^6 \ln m \log^2 (\frac{m\ n}{ah}) + \frac{m^4\ n \log m}{h} \log (\frac{m\ n}{ah}))$$

and, uniformly for all pairs I, h the expected execution time of the randomized version of the algorithm is

$$O(m^7 \ln m \log^2 (\frac{m\ n}{ah}) + \frac{m^4\ n \log m}{h} \log (\frac{m\ n}{ah}))$$

It follows that the function $T(m, n)$ used in the statements of our line bounds can be chosen as

$$m^8 \log m \log^2 n + m^4 n \log m \log n$$

7. Acknowledgments

The authors wish to thank David Johnson, Howard Karloff, Alexander Rinnooy Kan and Ron Shamir for their helpful comments.

References

[1] K. Eisemann, The trim problem, Management Science 3 (1957), pp. 279-284.

[2] W. Fernandez de la Vega and G.S. Lueker, Bin packing can be solved within $1 + \varepsilon$ in linear time, Combinatorica 1 (1981), pp. 349-355.

[3] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem, Operations Research 9 (1961), pp. 849-859.

[4] M. Grotschel, L. Lovasz and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981), pp. 169-198.

[5] D.S. Johnson, The NP-completeness column: an ongoing guide, Journal of Algorithms, September, 1982.