Discrete Applied Mathematics 22 (1988/89) 143–148
North-Holland

ONLINE VARIABLE-SIZED BIN PACKING*

Nancy G. KINNERSLEY and Michael A. LANGSTON
Department of Computer Science, Washington State University, Pullman, WA 99164-1210, USA

Received 25 February 1987
Revised 27 October 1987

The classical bin packing problem is one of the best-known and most widely studied problems of combinatorial optimization. Efficient offline approximation algorithms have recently been designed and analyzed for the more general and realistic model in which bins of differing capacities are allowed (Friesen and Langston (1986)). In this paper, we consider fast *online* algorithms for this challenging model. Selecting either the smallest or the largest available bin size to begin a new bin as pieces arrive turns out to yield a tight worst-case ratio of 2. We devise a slightly more complicated scheme that uses the largest available bin size for small pieces, and selects bin sizes for large pieces based on a user-specified fill factor $f \geq 1/2$, and prove that this strategy guarantees a worst-case bound not exceeding $1\,5 + f/2$

1. Introduction

In the classical bin packing problem, the objective is to pack a list of n pieces $P = \langle p_1, p_2, \ldots, p_n \rangle$, each with a size in the range $(0, 1]$, into the minimum number of unit-capacity bins. The general significance of this NP-complete problem is reflected in the great attention it has received in the literature (see [2] for an updated survey).

Recently, important generalizations of the bin packing problem have been investigated [3, 4] in which bin capacities may vary. In particular, the model of [4] permits a fixed collection of bin sizes, where the objective is to minimize the total space of the packing. This model is considerably more realistic than that of the classical problem. (We observe, for example, that the classical problem corresponds to a lumber yard that sells 2×4s in 8-foot lengths only!) In [4], some practical, offline algorithms for variable-sized bin packing were designed and analyzed. The most complicated of those, termed FFDLS, was proved always to produce a packing whose total space is asymptotically bounded by $\frac{4}{3}$ times the optimum. Also, from a more purely theoretical standpoint, an offline fully polynomial-time approximation scheme has very recently been devised in [11] using a linear programming formulation of the problem.

In this paper, we explore the worst-case behavior of fast, *online* variable-sized bin

*This research is supported in part by the National Science Foundation under grants ECS-8403859 and MIP-8603879, and by the Office of Naval Research under contract N-00014-88-K-0343

packing schemes. An online algorithm cannot preview and rearrange the elements of P before it starts to construct a packing, but must instead accept and immediately pack each piece as it arrives. A number of online strategies have been proposed and analyzed for the classical problem. See, for example, [9, 10, 12, 13]. What makes our problem even more difficult is that whenever an online algorithm decides to begin a new bin, it must also select the size of that bin, and cannot go back later to repack or consolidate bins.

2. Notation

Let k denote the number of distinct bin sizes available, where there is an unlimited supply of bins of each size. We normalize bin and piece sizes so that the largest bin is of size 1 (and thus the size of the largest piece cannot exceed 1). Let $B = \langle B_1, B_2, \ldots, B_l \rangle$ denote the ordered list of l bins containing P as packed by an online algorithm, ALG. We use B^* for the corresponding optimum packing with m bins. We employ the function s to specify bin and piece sizes, and use the function c to specify the total contents of a bin. For example, $s(p_1)$ denotes the size of the first piece and $c(B_1)$ denotes the cumulative size of all pieces ALG packs in its first bin. Finally, given an instance I of variable-sized bin packing, we use ALG(I) and OPT(I) to denote the values $\sum_{i=1}^{l} s(B_i)$ and $\sum_{i=1}^{m} s(B_i^*)$, respectively.

3. Some simple algorithms

One option for an online algorithm is simply to begin a new bin whenever the next available piece will not fit into the current bin. If bins of size 1 are always used, this O(n)-time scheme is called NFL (Next Fit, using Largest possible bins). The following result is from [4] and is reproduced here for the purpose of illustration.

Theorem 3.1. $\mathrm{NFL}(I) < 2 \cdot \mathrm{OPT}(I) + 1$ *for any instance* I.

Proof. For $1 \le i < l$, $c(B_i) + c(B_{i+1}) > 1$. Therefore

$$\sum_{i=1}^{l} c(B_i) > \tfrac{1}{2}(l-1)$$

and

$$\mathrm{NFL}(I) = (l-1) + 1 < 2 \sum_{i=1}^{l} c(B_i) + 1$$

$$= 2 \sum_{i=1}^{m} c(B_i^*) + 1 \le 2 \sum_{i=1}^{m} s(B_i^*) + 1 = 2 \cdot \mathrm{OPT}(I) + 1. \qquad \square$$

Any packing instance consisting of pieces of size $\frac{1}{2}+\varepsilon$ and bins of sizes 1 and $\frac{1}{2}+\varepsilon$, for some arbitrarily small $\varepsilon>0$, demonstrates that the bound of 2 is asymptotically tight for NFL.

Another alternative is to review all partially packed bins, placing the next available piece in the first bin with room for it, beginning a new bin only when necessary. If bins of size 1 are always used, we denote this approach by FFL (First Fit, using Largest possible bins). By efficiently conducting the review of partially packed bins [6], FFL can be implemented to run in $O(n \log n)$ time.

Theorem 3.2. $\mathrm{FFL}(I)<2\cdot\mathrm{OPT}(I)+1$ *for any instance* I.

Proof. Use the same series of arguments presented in the proof of Theorem 3.1. □

While the worst-case behavior of First Fit is superior to that of Next Fit for the classical problem [2, 5], this is not the case for NFL and FFL when applied to variable-sized bin packing. In fact, *any* online algorithm that uses the largest possible bins will produce the same packing when presented with a troublesome instance such as the one described immediately following the proof of Theorem 3.1.

Given the egregious behavior resulting from the use of large bins, we next consider FFS (First Fit, using Smallest possible bins), of time complexity $O(n \log n + n \log k)$.

Theorem 3.3. $\mathrm{FFS}(I)<2\cdot\mathrm{OPT}(I)+1$ *for any instance* I.

Proof. Since First Fit is used, $c(B_1)+c(B_l)>s(B_1)$. Also, $c(B_i)+c(B_{i+1})>s(B_i)$ for $1\le i<l$. Therefore,

$$\begin{aligned}
\mathrm{FFS}(I) &= \sum_{i=1}^{l} s(B_i) < s(B_1) + \sum_{i=1}^{l-1} s(B_i) + s(B_l) \\
&< 2 \sum_{i=1}^{l} c(B_i) + s(B_l) \\
&\le 2 \sum_{i=1}^{l} c(B_i) + 1 \\
&= 2 \sum_{i=1}^{m} c(B_i^*) + 1 \le 2 \sum_{i=1}^{m} s(B_i^*) + 1 \\
&= 2\cdot\mathrm{OPT}(I) + 1. \qquad \square
\end{aligned}$$

Unfortunately, FFS performs no better in the worst case than does NFL or FFL, since any packing instance consisting of pieces of size $\frac{1}{2}$ and bins of sizes 1 and $1-\varepsilon$, for some arbitrarily small $\varepsilon>0$, demonstrates that the bound of 2 is asymptotically tight for FFS.

4. Main result

We observe that FFL errs in its packing of "large" pieces (those with size exceeding $\frac{1}{2}$), while FFS errs in its packing of "small" pieces (those with size less than or equal to $\frac{1}{2}$). Therefore, we now focus our attention on a *hybrid* approach [7] that we shall denote by FFf. Let f denote a user-specified *fill factor* in the range $[\frac{1}{2}, 1]$. Suppose FFf must start a new bin using a piece p_i. If p_i is a small piece, then FFf starts a new bin of size 1. If p_i is a large piece, then FFf selects the smallest bin size in the range $[s(p_i), s(p_i)/f]$ if such a size exists, else it uses bin size 1. For example, if the fill factor is $\frac{3}{4}$ and a piece, p_i, needs a new bin, then FFf will select a unit-capacity bin if and only if either $s(p_i) \leq \frac{1}{2}$ or there is no bin with size less than 1 available that p_i can fill at least $\frac{3}{4}$ full.

Theorem 4.1. $\mathrm{FFf}(I) < (1.5 + \frac{1}{2}f) \cdot \mathrm{OPT}(I) + 2$ *for any instance* I.

Proof. Given an arbitrary instance, I, we classify bins of the FFf packing as follows: a bin of type X has size 1 and contains a single piece; a bin of type Y has size 1 and contains two or more pieces; a bin of type Z has size less than 1.

We deviate from this classification for at most two "exceptional" bins. Every bin of type X, except at most one, must contain a large piece. (To see this, observe that if a bin of type X contains a small piece, then every subsequent bin of type X must contain a large piece.) If there is a bin of type X that contains a small piece, then we change its classification from type X to exceptional. Similarly, every bin of type Y, except at most one, must be more than $\frac{2}{3}$ full. (To see this, observe that if a bin of type Y is at most $\frac{2}{3}$ full, then every subsequent bin of type Y must contain at least two pieces, each of size greater than $\frac{1}{3}$.) If there is a bin of type Y that is at most $\frac{2}{3}$ full, then we change its classification from type Y to exceptional.

Let x and y denote the number of bins of types X and Y, respectively. Let z denote the sum of the sizes of all bins of type Z. Thus $\mathrm{FFf}(I) = x + y + z + s$ (any exceptional bins) and we have

$$\mathrm{FFf}(I) - 2 \leq x + y + z. \tag{1}$$

We now obtain two distinct lower bounds for $\mathrm{OPT}(I)$. For the first, we consider the bin sizes available. Since every bin of type X or Z contains a piece whose size exceeds $\frac{1}{2}$, no two such pieces can share one bin in an optimal packing of I. From the way FFf selects a new bin for a large piece when one is required, a piece, p_i, from a bin of type X requires a bin of size at least $\min\{1, s(p_i)/f\} \geq 1/(2f)$ in an optimal packing. Also, any large piece used by FFf to begin a bin of type Z is packed as tightly as possible. Therefore, we have

$$\mathrm{OPT}(I) \geq \tfrac{1}{2}x/f + z. \tag{2}$$

For the second lower bound, we consider the total size of all pieces. The contents of type X bins sum to more than $\frac{1}{2}x$. The contents of type Y bins sum to more than

$\frac{2}{3}y$. The contents of type Z bins sum to at least fz. Thus, since $x+y>0$, we have

$$\mathrm{OPT}(I) > \tfrac{1}{2}x + \tfrac{2}{3}y + fz. \tag{3}$$

Let R denote $(\mathrm{FFf}(I)-2)/\mathrm{OPT}(I)$. From (1) and (2) we know

$$R \leq (x+y+z)/(\tfrac{1}{2}x/f+z)$$

from which we derive

$$y \geq \tfrac{1}{2}Rx/f + Rz - x - z. \tag{4}$$

From (1) and (3) we know

$$R < (x+y+z)/(\tfrac{1}{2}x+\tfrac{2}{3}y+fz)$$

in which we substitute the lower bound for y from (4), since it is known [1, 8] that, even for unit-capacity bins, *any* online algorithm's worst-case ratio exceeds $1.536 > \frac{3}{2}$. Thus we derive

$$R < (3x+fx+fz(10-6f))/(2x+4fz)$$

which is bounded above by $\frac{1}{2}(3x+fx)/x = 1.5+\frac{1}{2}f$ as long as R is bounded below by $\frac{7}{4}$. Therefore,

$$\mathrm{FFf}(I) = R\cdot\mathrm{OPT}(I)+2 < (1.5+\tfrac{1}{2}f)\cdot\mathrm{OPT}(I)+2. \qquad \square$$

5. Discussion

Surprisingly, we conclude from Theorem 4.1 that the simplest variant of FFf may be the best (in the worst-case sense). By setting $f=0.5$, a small piece needing a new bin always gets the largest bin available while a large piece needing a new bin always gets the smallest bin that can contain it. Let FFH denote this limiting-case hybrid.

Corollary 5.1. $\mathrm{FFH}(I) < 1.75\cdot\mathrm{OPT}(I)+2$ *for any instance* I.

For the classical problem, examples exist [5] to demonstrate that $\mathrm{FF}(I)$ can approach arbitrarily close to $1.7\cdot\mathrm{OPT}(I)$ from below. Therefore, of course, the same holds for FFH under the packing model addressed herein.

The determination of just how tight the bounds given in Theorem 4.1 are is as yet an open issue. (Slightly more complicated arguments easily reduce the additive constant from 2 to 1.) However, examples such as the one that follows demonstrate that Corollary 5.1 does not extend to arbitrary values of f, and that the guarantee of Theorem 4.1 is indeed dependent on f. Let n be evenly divisible by 4. Suppose that P contains $\frac{1}{2}n$ pieces of size $\frac{1}{3}+\varepsilon$ followed by $\frac{1}{2}n$ pieces of size $\frac{1}{2}+\varepsilon$, and that bin sizes are $\frac{5}{6}+2\varepsilon$ and 1 for some arbitrarily small $\varepsilon>0$. Thus, for $f=0.6$, $\mathrm{FFf}(I)$ can exceed any value strictly less than $1.8\cdot\mathrm{OPT}(I)$.

Finally, let the parameter q denote an integer greater than 1 and suppose $s(p_i) \leq 1/q$ for $1 \leq i \leq n$. (In this case, FFf reduces to FFL.) As in [5, Theorem 2.3] we note that FFL insures the inequality $c(B_i) > q/(q+1)$ for all but at most two values of i in the range $[1, l]$. Therefore, in this parameterized environment, there is no worst-case penalty for variable-sized bin packing. That is, we have

$$\mathrm{FFL}(I) < ((q+1)/q) \cdot \mathrm{OPT}(I) + 2,$$

the same as for the classical problem. However, the freedom to choose bin (and piece) sizes of $1/(q+1)+\varepsilon$, for arbitrarily small $\varepsilon > 0$, greatly simplifies the job of establishing the asymptotic tightness of this parameterized bound.

References

[1] D J Brown, A lower bound for on-line one-dimensional bin packing algorithms, Tech Rept R-864, Coordinated Science Laboratory, University of Illinois, Urbana, IL (1979)

[2] 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, eds , Algorithm Design for Computer Systems Design (Springer, New York, 1984), 49–106

[3] D K Friesen and M A Langston, A storage size selection problem, Inform Process Lett 18 (1984) 295–296

[4] D K Friesen and M A Langston, Variable sized bin packing, SIAM J Comput 15 (1986) 222–230

[5] 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 (1974) 299–325

[6] D S Johnson, Fast algorithms for bin packing, J Comput Syst. Sci 8 (1974) 272–314.

[7] M A Langston, A study of composite heuristic algorithms, J Oper Res Soc 38 (1987) 539–544.

[8] F M Liang, A lower bound for on-line bin packing, Inform Process Lett 10 (1980) 76–79

[9] C C Lee and D T Lee, A simple on-line bin packing algorithm, J. ACM 32 (1985) 562–572

[10] C Martel, A linear on-line bin packing algorithm, Tech Rept CSRL-83-12, Department of Electrical and Computer Engineering, University of California, Davis, CA (1983).

[11] F D Murgolo, An efficient approximation scheme for variable-sized bin packing, SIAM J Comput 16 (1987) 149–161

[12] P Ramanan and D J Brown, On-line bin packing in linear time, Tech Rept R-1003, Coordinated Science Laboratory, University of Illinois, Urbana, IL (1983)

[13] A C Yao, New algorithms for bin packing, J ACM 27 (1980) 207–227