Control and Cybernetics

vol. **43** (2014) No. 4

Bin packing with restricted item fragmentation*†

by

Krzysztof Pieńkosz

Institute of Control and Computation Engineering,
Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
K.Pienkosz@ia.pw.edu.pl

Abstract: In this paper we consider a generalization of the bin packing problem, in which it is permitted to fragment the items while packing them into bins. There is, however, a restriction that the size of each piece of the fragmented item cannot be smaller than a given parameter β. An interesting aspect of such a model is that if $\beta = 0$, then the problem can be easily solved optimally. If β is large enough, meaning, in fact, that the fragmentation is not allowed, we get the classical bin packing problem, which is *NP*-hard in the strong sense. We present approximation algorithms for solving the problem and analyse their properties. The results of computational experiments and conclusions relating to the effectiveness of the algorithms are also presented.

Keywords: bin packing, fragmentable items, approximation algorithms

1. Introduction

Bin packing is the problem, in which a given set of items of various sizes should be packed into the minimum number of bins of identical capacity. The common assumption is that items packed may not be fragmented into smaller pieces. There are, however, several applications in which this requirement is too restrictive (see, e.g. Mandal, Chakrabarti and Ghose, 1998; Namman and Rom, 2001; Shachnai, Tamir and Yehezkely, 2008; Epstein and van Stee, 2008). By packing the items in pieces we can reduce the number of necessary bins and thus more efficiently utilize the resources available. In practical applications some restrictions on item fragmentation usually exist, due to technological or economic reasons. The typical requirement is that each piece of a fragmented item must be reasonable in size, i.e. it cannot be smaller than a given parameter

*The original version of this paper was presented at the BOS 2014 (Operations and Systems Research) Conference in Warsaw, September 2014.

†Submitted in August 2014, Accepted in November 2014.

β. We will call such situation the *bin packing problem with restricted item fragmentation*. The motivation for studying this kind of problem comes from the specific application where it is required to satisfy the demands for iron bars of various lengths. The bars can be obtained by cutting them entirely from stock or welding from smaller pieces. Because of technological reasons the pieces of bars which are welded together cannot be too small.

The literature provides also examples of other variants of bin packing models with item fragmentation. Thus, Mandal, Chakrabarti and Ghose, (1998), Menakerman and Rom (2001), and Namman and Rom (2001) considered the problems, in which there is no restriction on the minimum size of the pieces, but fragmentation is associated with additional consumption of bin capacity. The authors mentioned showed that these problems are *NP*-hard and Shachnai and Yehezkely (2007), Shachnai, Tamir and Yehezkely (2008) developed approximation schemes for them. In Menakerman and Rom (2001) and Shachnai and Yehezkely (2007), Shachnai, Tamir and Yehezkely (2008), the costs of item fragmentations were considered and algorithms for bin packing with minimal or limited number of fragmentations were proposed. Another bin packing model with item fragmentation was studied by Epstein and van Stee (2008). They designed approximation schemes for the problem where items may be fragmented arbitrarily, but there is an upper bound on the number of parts permitted to be packed into each bin.

The remainder of the paper is organized as follows. In Section 2 we formally define the bin packing problem with restricted item fragmentation. Sections 3, 4 and 5 are devoted to the analysis of the three classes of the proposed approximation algorithms. In Section 6 the computational results are provided to illustrate the effectiveness of the algorithms for various values of the parameter β and concluding remarks are presented.

2. Problem formulation

The bin packing problem with restricted item fragmentation can be formulated as follows:

Problem BPF_β

Given a set N of n items, each having size S_j, pack them into the minimum number of bins of identical capacity C. Items may be fragmented into smaller pieces provided that the size of each piece is at least β.

We assume that $C \geqslant S_j > 0$ for all $j \in N$ and we treat β as the parameter of the model. In our formulation the original sizes of items may be $S_j < \beta$ for some $j \in N$, but the pieces resulting from fragmentation (of other items) cannot be smaller than β. It follows from the formulation of the problem that an item j for which $S_j < 2\beta$ cannot be split since the sizes of both parts should be at least β. Further on, we will call such items *undividable*, whereas the remaining items will be referred to as *fragmentable*.

It turns out that computational complexity of Problem BPF_β strongly de-

pends on the value of the parameter β. If $\beta > C/2$ no item can be fragmented, and so we get the classical bin packing problem which is *NP*-hard in the strong sense (see, e.g. Garey and Johnson, 1979). It is a well known combinatorial problem and many approximation and exact algorithms have been developed for solving it, see e.g. Johnson et al. (1974), Fleszar and Charalambous (2011), Scholl, Klein and Jurgens (1997).

On the other hand, if $\beta = 0$, then all items are fragmentable and the problem is very easy to solve. We can pack items successively into bins until the first item, which does not fit into the current bin, is found. This item is fragmented into two parts. The first part fills up the bin, while the second part is packed into a new one. In this way we get an optimal solution with the following number of bins

$$LB = \left\lceil \sum_{j \in N} S_j / C \right\rceil \tag{1}$$

where symbol $\lceil \rceil$ denotes rounding up to the nearest integer.

In the following sections we present algorithms for the bin packing problems with arbitrary values of β. Note that LB from formula (1) may be treated as a lower bound on the optimal number of bins for these problems.

When presenting the algorithms we will denote them by $NAME_\beta$ if a given algorithm applies the fragmentation of items. The special variant of such algorithm, in which fragmentation is not allowed, will be denoted by *NAME* (without subscript β). Analogously, for a given instance I of the bin packing problem, $NAME_\beta(I)$ and $NAME(I)$ stand for the number of bins used by the algorithm $NAME_\beta$ and its variant *NAME*, respectively. We also use $OPT_\beta(I)$ to denote the optimal number of bins for packing problem with restricted item fragmentation.

3. Item oriented algorithms

Since the classical bin packing problem is *NP*-hard in the strong sense, approximate algorithms have been studied extensively. The best known heuristics are based on the *FF* (*First-Fit*) and *BF* (*Best-Fit*) concept (see Johnson et al., 1974). They start with empty bins, consider the items in a specified order and try to find place for each of them in one of the bins.

Algorithm *FF* puts the item into the bin with the smallest index that has sufficient residual capacity. Algorithm *BF* puts the current item into the bin with the smallest but sufficient residual capacity. Ties are broken in favour of the lowest indexed bin.

Although these algorithms are based on very simple strategies, they often provide satisfactory results, especially if items are considered in non-increasing order of their sizes, i.e. the item with maximum size is packed first. Such variants of algorithms are called *FFD* (*First-Fit Decreasing*) and *BFD* (*Best-Fit Decreasing*).

To adapt these algorithms to the model with restricted item fragmentation let us note that a given item j may be put into a bin either in full size or as a fragment of size at least β and not greater than $S_j - \beta$. This condition follows from the fact that the sizes of both parts of each fragmented item must be at least β.

In the description of algorithms we assume that s_j denotes the current size of item j and c_i the current residual capacity of bin i. At the beginning we have $s_j = S_j$ and $c_i = C$ for all items and bins. Let N_F be a set of fragmentable items, i.e. $N_F = \{j \in N : S_j \geqslant 2\beta\}$. The first packing strategy for Problem BPF_β is as follows.

Packing Rule R1

- If $s_j \leqslant c_i$ pack the whole item j into bin i.
- If $s_j > c_i \geqslant \beta$ and $j \in N_F$ then split item j and pack only fragment of size $\min\{s_j - \beta, c_i\}$ into bin i.

The algorithms using this strategy are referred to as FF_β and BF_β.

Algorithm FF_β

1. Create a list L of items.
2. Take the first item j from L.
3. Find the lowest indexed bin for which $s_j \leqslant c_i$ or $c_i \geqslant \beta$ if $j \in N_F$.
4. Apply packing rule R1. If item j is fragmented, then insert the remaining piece into list L.
5. If list L is not empty, go to Step 2, else STOP.

Algorithm BF_β differs from FF_β only in Step 3. It chooses the bin whose residual capacity would be the smallest if packing rule R1 is applied. If there is more than one such bin, the one with the smaller index is preferred.

In algorithms FF_β and BF_β no specific ordering of items in the list L is assumed. The variants of FF_β and BF_β where items are ordered according to non-increasing sizes are referred to as FFD_β and BFD_β. The properties of these algorithms are as follows.

Proposition 1. *For any instance I of Problem BPF_β and any ordering of items $FF_\beta(I) \leqslant \lceil R(\beta)OPT_\beta(I)\rceil$ and $BF_\beta(I) \leqslant \lceil R(\beta)OPT_\beta(I)\rceil$, where*

$$R(\beta) = \begin{cases} \frac{C}{C-2\beta} & for\ 0 \leqslant \beta < C/4 \\ 2 & for\ \beta \geqslant C/4. \end{cases}$$

Proof. Note, that in any solution provided by algorithms FF_β and BF_β only one bin can have at least $C/2$ space left. To see this, suppose that there are two such bins, say k and l, where $k < l$. Then we come to the conclusion that items from bin l fit in bin k, so algorithms FF_β and BF_β would pack them into lower indexed bin k instead of l. Let z be the number of bins used by FF_β or BF_β. Since all bins except at most one are more than half full then

$$\sum_{j\in N} S_j > (z-1)C/2$$

and

$$z < 2\frac{\sum_{j\in N} S_j}{C} + 1 \leqslant 2\,OPT_\beta(I) + 1 \quad \text{so} \quad z \leqslant \lceil 2OPT_\beta(I) \rceil \ .$$

For $\beta < C/4$ we can strengthen the above upper bound on z if we notice that there can be at most one bin with residual capacity at least 2β. In case of algorithm FF_β, if $c_k \geqslant 2\beta$ for some bin k and there are still items in the list L, then they are first put entirely or partially into bin k before filling a new bin. In algorithm BF_β an item can be put into a new bin instead of k only if it is greater than $C - \beta$, so both algorithms fill at most one bin to capacity no larger than $C - 2\beta$. Hence

$$\sum_{j\in N} S_j > (z-1)(C-2\beta)$$

and

$$z < \frac{\sum_{j\in N} S_j}{C-2\beta} + 1 = \frac{C}{C-2\beta}\frac{\sum_{j\in N} S_j}{C} + 1 \quad \text{so} \quad z \leqslant \left\lceil \frac{C}{C-2\beta} OPT_\beta(I) \right\rceil \ . \qquad \square$$

It follows from Proposition 1 that if $\beta = 0$ then $FF_\beta(I) \leqslant OPT_\beta(I)$ and $BF_\beta(I) \leqslant OPT_\beta(I)$, so both algorithms give an optimal solution. The computational experiments show, however, that in many other cases the performance of algorithms FF_β and BF_β is poor. There are even instances of bin packing problems, for which algorithm *FFD* without applying item fragmentation gives a better solution than FFD_β and BFD_β.

Example 1. Consider the packing problem with 6 items of sizes 6, 6, 3, 3, 2 and 2. The capacity of bins is 11 and $\beta = 2$. Algorithms *FFD* and *BFD* pack the first, third and the fifth item into bin 1, filling it completely, and the remaining items into bin 2. Two bins are used. Algorithms FFD_β and BFD_β pack the first item into bin 1. The second item is fragmented and only four units of it are put into bin 1 since the size of the second fragment must be at least 2. Items 3, 4, the remaining piece of item 2 and item 5 are packed into the second bin consuming 10 units of its capacity. The last item must be packed into the third bin.

4. Bin oriented algorithms

The algorithms presented in the previous section are item oriented. It means that for each particular item they try to find a bin, in which this item could be placed. We can also propose, though, the bin oriented heuristics, which consider bin by bin. They start with an empty bin, open it and attempt to fill it as much as possible considering items from the list L and applying the packing rule R1. When no item or its fragment fits in the current bin, it is closed and the next bin is opened. This process is repeated until all items are packed.

We call such heuristics *BIN-FF*$_\beta$. The special variant is *BIN-FFD*$_\beta$, where items are ordered in the list L according to non-increasing sizes. It turns out

that the results produced by the algorithms *BIN-FF*$_\beta$ and *FF*$_\beta$ are very often similar. This fact is partially justified by the following properties.

Proposition 2. *If for a given instance I of bin packing problem without fragmentation the ordering of the list L is the same in both algorithms BIN-FF and FF, then BIN-FF*(I) = *FF*(I).

Proof. In both algorithms each bin is filled with those items which do not fit in the previous bins. They are packed in the same sequence resulting from ordering of the list L. □

Proposition 3. *BIN-FFD*$_\beta(I)$ = *FFD*$_\beta(I)$ *for any instance I of Problem BPF*$_\beta$.

Proof. If items are not fragmented, the property follows from Proposition 2. So, let k be the first item that is fragmented. Proposition 2 implies that all items preceding k in the list L are packed in the same way by both algorithms, so item k must be also fragmented in the same way. Because the ordering of the list L is non-increasing, the remaining part of item k is inserted in the same position of the list L in both algorithms. So, the packing order of all fragmented pieces is the same in both algorithms and according to Proposition 2 the number of used bins must be equal. □

More effective is usually the algorithm *BIN-BF*$_\beta$. It also packs bin by bin but selects items in a different way. The priority is given to the item that fills the bin maximally, i.e. after packing it with rule R1 the residual capacity in the bin is the smallest possible. The ordering of the list L affects the algorithm *BIN-BF*$_\beta$ slightly and only when fragmentable items are considered.

Proposition 4. *For any instance I of the bin packing problem without fragmentation BIN-BF*(I) = *BIN-FFD*(I) = *FFD*(I).

Proof. To fill the bin as much as possible, the algorithm *BIN-BF* always selects the largest item that fits, so *BIN-BF*(I) = *BIN-FFD*(I). Next, it follows from Proposition 2 that *BIN-FFD*(I) = *FFD*(I). □

5. Item grouping

An essential improvement in bin packing with restricted item fragmentation may be achieved if we analyse the structure of items and reserve big fragmentable items mainly for completing the filling of the bins. Starting with an empty bin we can first attempt to pack the undividable items. They should consume no more than $C - \beta$ units of bin capacity in total, so that remaining space could be filled up with a fragmentable item. The packing rule for such strategy is a little more complicated and it looks as follows.

Packing Rule R2

- If $s_j = c_i$ or $s_j \leqslant c_i - \beta$ then pack the whole item j into bin i.
- If $s_j \in (c_i - \beta, c_i)$ and $j \in N_F$ then split item j and pack only the fragment of the size $s_j - \beta$.

- If $s_j \in (c_i,\ c_i + \beta)$ and $j \in N_F$ and $c_i \geqslant 2\beta$ then pack the fragment of the size $c_i - \beta$.
- If $s_j \geqslant c_i + \beta$ and $j \in N_F$ then pack the fragment of the size c_i.
- Otherwise do not pack item j into bin i.

Packing rule R2 ensures that bins are either completely filled up or have residual capacity not smaller than β. It is advantageous to use such rule as long as fragmentable items are still available in the list L. If there is no fragmentable item then packing rule R1 should be rather applied. The following conditions are proposed to specify the situations when rule R2 should be replaced by rule R1.

C1.– all the remaining items in the list L are undividable.
C2.– the considered item is undividable and is greater than $C - \beta$.
C3.– the residual capacity of bin is $c_i < 2\beta$ and the sizes of all items in the list L are in the range $(c_i - \beta,\ c_i + \beta)$.

Conditions C1 and C2 are rather obvious. Condition C3 relates to such situations when although some fragmentable items are still in the list L, they are too small to fill up the bin completely, so it is no use reserving capacity for them. If any of conditions C1, C2 or C3 is met, we propose to continue packing the bin with *BIN-BF*$_\beta$. The scheme of the whole algorithm may be summarized as follows.

Algorithm *BIN-FFSL*$_\beta$

1. Create the list L with undividable items ordered according to non-increasing sizes and followed by fragmentable items ordered in non-decreasing way.
2. Take next empty bin.
3. Apply the packing rule R2 to successive items from the list L until the bin is full or one of conditions C1, C2 or C3 is satisfied. Insert the pieces resulting from fragmentation into the list L according to its ordering.
4. If any of conditions C1 or C2 or C3 is satisfied, complete the packing of the bin with the algorithm *BIN-BF*$_\beta$.
5. If the list L is not empty, go to Step 2, else STOP.

Example 2. Consider the instance from Example 1. Algorithm *BIN-BF*$_\beta$ results in identical packing as *FFD*$_\beta$ and *BFD*$_\beta$ using 3 bins. Algorithm *BIN-FFSL*$_\beta$ reorders items so that items 3, 4, and 5 are packed first. Then, item 1 is fragmented and 3 units are packed to fill up the first bin. The remaining fragment of this item and the other items are packed into the second bin.

Algorithm *BIN-FFSL*$_\beta$ is bin oriented and has interesting properties. It can be verified that Proposition 1 is also valid for it. If all items are undividable, the *BIN-FFSL*$_\beta$ algorithm works as *BIN-BF*$_\beta$, which is in this case equivalent to *FFD*, according to Proposition 4. On the other hand, if items are big and fragmentable, the algorithm *BIN-FFSL*$_\beta$ gives optimal solutions.

Proposition 5. *If $S_j \geqslant 3\beta$ for all $j \in N$ then algorithm BIN-FFSL$_\beta$ provides an optimal solution using LB bins.*

Proof. First, let us note that when starting packing a new bin, there may be at most two fragmented items that are smaller than 3β, and both of them cannot have sizes $s_j \in (\beta, 2\beta)$. Items smaller than 3β are packed into the bin first. Now, if all the remaining items fit in that bin, they are packed there. If they exceed its residual capacity but less than β units then the bin remains unfilled and the next bin is used. Otherwise, the packing rule R2 ensures that the bin is filled up completely and again at most two new fragments appear in the list L. So, when the algorithm *BIN-FFSL*$_\beta$ stops, the slack capacity may be only in the last bin and in the previous to the last one. However, the number of used bins denoted by z always satisfies the condition

$$(z-1)C < \sum_{j \in N} S_j \leqslant zC$$

so $z = LB$, which is the lower bound on the optimal number of bins. □

It follows from Proposition 5 that if in Problem *BPF*$_\beta$ all items have the size of at least 3β, then the optimal number of bins is *LB*. Note that the computational complexity of the algorithm *BIN-FFSL*$_\beta$ is $O(|N|^2)$. So, another important conclusion is that if $\beta \leqslant \min\{S_j/3\colon\ j \in N\}$ then Problem *BPF*$_\beta$ may be solved optimally in polynomial time. In general case, this problem is *NP*-hard in the strong sense as it is a generalization of the classical bin packing problem.

6. Computational results

The performance of the presented algorithms has been tested using the data instances considered in Scholl, Klein and Jurgens (1997). There are 36 groups of 20 instances with 50, 100, 200 and 500 items. Bin capacity is 100, 120 or 150 and the sizes of items are integer values generated randomly from the ranges [1, 100], [20, 100] and [30, 100].

We have compared algorithms *FFD* and *BFD* assuming that fragmentation is not exploited and algorithms *FF*$_\beta$, *BF*$_\beta$, *BIN-FF*$_\beta$, *BIN-BF*$_\beta$ and *BIN-FFSL*$_\beta$ for various values of parameter β. It turns out that algorithms *FFD* and *BFD* give almost identical results. In only one case out of 720 tests *BFD* has used one bin less than *FFD*. The performance of *FFD*$_\beta$ and *BFD*$_\beta$ is also similar. The algorithm *BFD*$_\beta$ has been slightly better in 25 cases for some values of β. For the algorithm *BIN-BF*$_\beta$ a variant with non-decreasing list L seems to be the most advantageous and we denote it by *BIN-BFI*$_\beta$.

Figures 1 and 2 illustrate the performance of algorithms *BFD*, *BFD*$_\beta$, *BIN-BFI*$_\beta$ and *BIN-FFSL*$_\beta$ for various values of β. The data sets are N2C2W1 and N2C2W2, where 100 items are considered and capacity of bins is 120. The sizes of items are between 1 and 100 for N2C2W1 and between 20 and 100 for N2C2W2. The results shown in Figs. 1 and 2 represent the average number of used bins over 20 instances of each data set.

Computational experiments show that algorithms *FFD*$_\beta$ and *BFD*$_\beta$ are outperformed. In many cases they give worse results than *FFD* or *BFD* which do

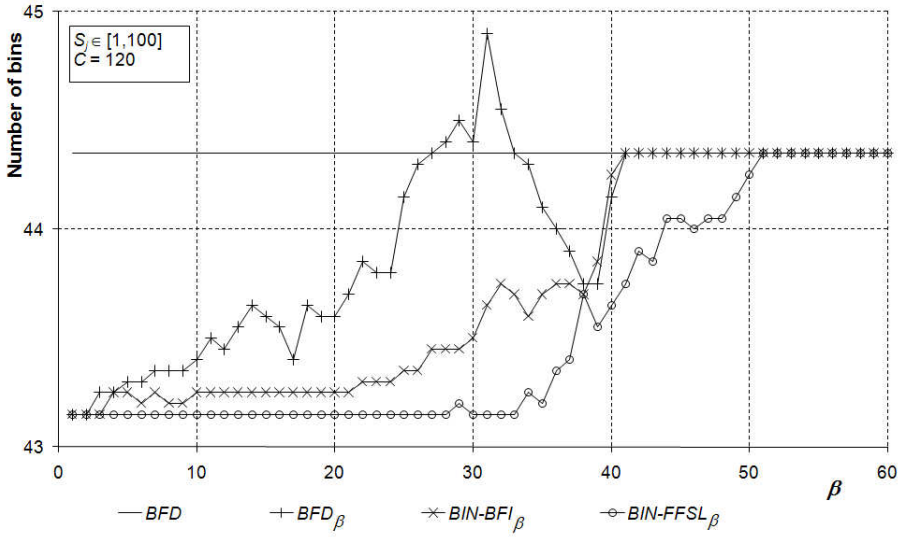

Figure 1. The average number of bins used by the algorithms for the data set N2C2W1 and various values of β

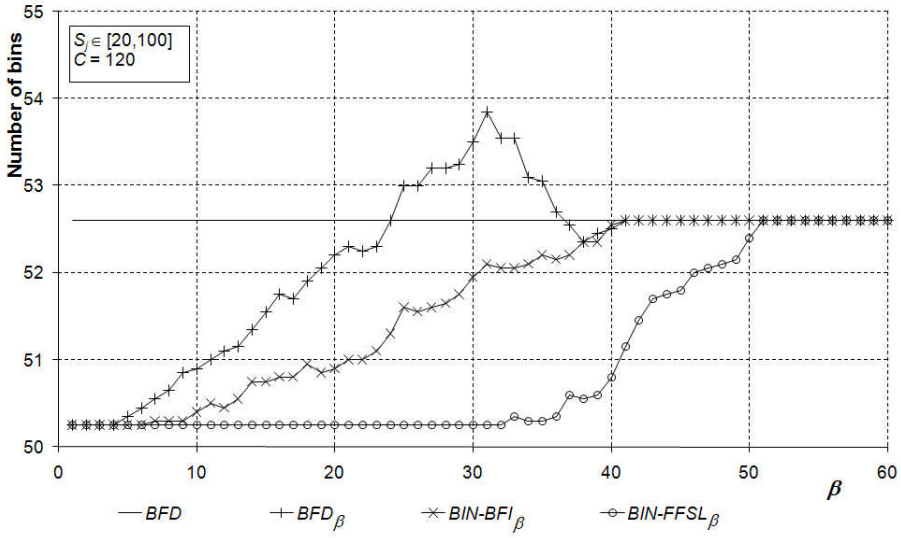

Figure 2. The average number of bins used by the algorithms for the data set N2C2W2 and various values of β

not exploit fragmentation. A much better algorithm than FFD_β and BFD_β turns out to be $BIN\text{-}BFI_\beta$. However, the most effective is $BIN\text{-}FFSL_\beta$ which gives optimal solutions for a wide range of β values. Proposition 5 specifies some sufficient conditions for such cases. The crucial role in effective packing is played by the big fragmentable items with sizes of at least 3β. In practice, a relatively small number of such items allows to limit utilization of bins to the level close to the lower bound LB. It is worth noting that for both data sets N2C2W1 and N2C2W2 there is no item with size exceeding 3β when $\beta \geqslant 34$. If we look at Figs. 1 and 2 we can notice that $\beta = 34$ is also the threshold value, above which the number of used bins starts to increase drastically for the algorithm $BIN\text{-}FFSL_\beta$.

References

Epstein L. and van Stee R. (2008) Approximation schemes for packing splittable items with cardinality constraints. *Lecture Notes in Computer Science*, **4927**, 232-245.

Fleszar K. and Charalambous C. (2011) Average–weight–controlled bin–oriented heuristics for the one–dimensional bin–packing problem. *European Journal of Operational Research*, **210**, 176-184.

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

Johnson D.S., Demers A., Ullman J.D., Garey M.R., Graham R.L. (1974) Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, **3**, 299-325.

Mandal C.A., Chakrabarti P.P., Ghose S. (1998) Complexity of fragmentable object bin packing and an application. *Computers and Mathematics with Applications*, **35**, 91-97.

Menakerman N. and Rom R. (2001) Bin packing with item fragmentation. *Lecture Notes in Computer Science*, **2125**, 313-324.

Namman N. and Rom R. (2001) Analysis of transmission scheduling with packet fragmentation. *Discrete Mathematics and Theoretical Computer Science*, **4,** 139-156**.**

Scholl A., Klein R., Jurgens C. (1997) BISON: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, **24**, 627-645.

Shachnai H. and Yehezkely O. (2007) Fast asymptotic FPTAS for packing fragmentable items with costs. *Lecture Notes in Computer Science*, **4639**, 482-493.

Shachnai H., Tamir T., Yehezkely O. (2008) Approximation schemes for packing with item fragmentation. *Theory of Computing Systems*, **43**, 81-98.