Journal of Mathematical Modelling and Algorithms (2006) **5:** 353–370
DOI: 10.1007/s10852-005-9025-y

Models and Complexity of Multibin Packing Problems

P. LEMAIRE★, G. FINKE and N. BRAUNER
Laboratoire Leibniz – IMAG, 46 avenue Félix Viallet, 38031, Grenoble cedex, France.
e-mail: {pierre.lemaire, gerd.finke, nadia.brauner}@imag.fr.

(Received 21 October 2004; in final form 11 October 2005)

Abstract. This paper extends the previous work on multibin packing problems and gives a deeper insight into these models and their complexity, so as to provide a strong framework for future application-oriented studies. In a multibin problem, an object requires several bins to be packed. New models are represented, including the maximum cardinality multibin packing. Their complexity is studied and several pseudo-polynomial time algorithms are described, together with a fully polynomial time approximation scheme (FPTAS) for a fixed number of bins.

Mathematics Subject Classifications (2000): 90B30, 90B35, 05B40.

Key words: multibin packing, complexity, FPTAS.

The packing of multibin objects is a concept recently introduced [21, 22]. In this paper, we extend the existing models, classification, and complexity results. The purpose is not to provide fast and practically efficient methods to solve some particular cases. In fact, we aim at providing the theoretical framework on which such procedures may be built. We think our results are also valuable for practitioners who are confronted with this type of bin-packing problems.

In Section 1, we present multibin problems, with a classification and some illustrative applications. Section 2 underlines the link between multibin packing and multiprocessor-task scheduling. In Section 3, we deal with the complexity of multibin packing problems and present some polynomial cases as well as pseudo-polynomial algorithms that apply to a fixed number of bins. Section 4 is dedicated to a fully polynomial time approximation scheme (FPTAS), that is an $(1+\epsilon)$-approximation algorithm that runs in time polynomial both in the instance size and in $1/\epsilon$.

★ Corresponding author.

1. Multibin Packing

The concept of multibin packing extends several well-known problems such as multiprocessor scheduling, bin-packing and maximum cardinality bin-packing, depending on the given objective.

In a classical packing problem, one has n objects and m identical bins. If one wants to pack all objects into a minimum number of bins of fixed height, one obtains the bin packing problem [10]. If one wants to pack the objects (tasks) in a fixed number of bins (processors) of minimal size, then it is a multiprocessor scheduling problem [5, 9]. If one wants to pack as many objects as possible into a given number of fixed height bins, one has a maximum cardinality bin packing problem [17, 18].

For those three problems, an object requires space in only one bin. An extension is the multibin packing. Indeed, a multibin problem consists of m bins and n objects. Each object is made of several identical parts, and all parts of an object must be packed into different bins. Note that the number of parts may differ from object to object, and that the order of the objects within a bin is of no importance.

This generalization is very close to multiprocessor-task scheduling, where a task requires several processors in parallel for its execution [6, 11]. Multibin packing is a relaxation of this latter problem: the constraint that the different parts of a task are performed in parallel is dropped (e.g., consider the Gantt chart of a multiprocessor-task schedule, turn it by 90° and let the tasks drop: the result is a multibin packing; see Figure 1). Multibin packing is also linked to scheduling (or bin packing) with conflicts [3, 8, 14, 15], where a task (object) is made of only one part, but some tasks (objects) cannot share the same processor (bin).

1.1. CLASSIFICATION

Several models may be considered, depending on the type of objects and bins. Because of their similarities, we use the notations and classification for multibin problems similar to the ones used for multiprocessor-task problems [6, 11]. A multibin problem is described by a three-field notation: $\alpha|\beta|\gamma$.

The first field, α, describes the bins. $\alpha = B$ means that all bins are identical. Some additional constraints may be given: the number m of bins may be fixed (e.g., $\alpha = Bm$), and the height H of the bins may be fixed as well (e.g., $\alpha = B^H$).

The second field, β, describes the set of objects. As for multiprocessor problems, we distinguish:

- size_j: each object O_j requires a fixed number size_j of bins and occupies a height h_j in each of them.
- any_j: each object O_j may be put into any number of bins, and its height $h_j(\text{size}_j)$ depends on the number size_j of bins it occupies. This is a generalization of the size_j model.

- fix_j: each object O_j must be put into a fixed set of bins. (This case has no real interest in the case of multibin packing, since one has no choice and each feasible placement is optimal.)
- set_j: each object O_j can be put into several alternative sets of bins, and it occupies a height h_j in the chosen ones. This is a generalization of the fix_j model.

Besides these specifications of the model, one can restrict the heights and widths (i.e., the number of bins to be used) of the objects:

- $h_j = 1$: restriction to objects of height 1.
- $h_j \leq k$: restriction to objects of height less than or equal to k.
- $\text{size}_j = k$: restriction to objects of width k.
- $\text{size}_j \leq k$: restriction to objects of width less than or equal to k.

The third field, γ, is the objective. Because the order of the objects within the bins is of no importance, some criteria for multiprocessor scheduling have no meaningful equivalents for multibin packing. The three main criteria are:

- $H_{\max}$: minimize the maximum height of m bins;

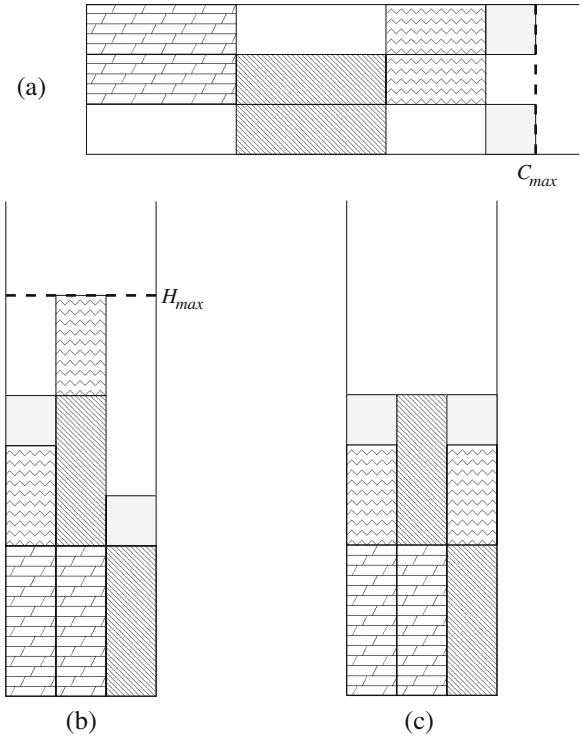

Figure 1. (a) A multiprocessor schedule of optimal schedule length $C_{\max}$; (b) a multibin packing of value $H_{\max}$; (c) an optimal packing.

- M: minimize the number of bins of height H to be used;
- N (or W): maximize the number (or weighted number) of objects packed into m bins of height H.

1.2. SOME ILLUSTRATIVE APPLICATIONS

There exist many applications of special cases of multibin packing problems. In this subsection, we list some of them.

1.2.1. *Telecommunication Network*

A first application of multibin problems arises in the design of bidirectional SONET/SDH networks [27, 28]. Once a SONET/SDH ring has been selected with traffic demands $d_{i,j}$ between pairs of its nodes, each of these demands must be routed in one of the two possible ways around the ring. The objective is to minimize the maximum load on the ring (the load of an edge is the sum of the demands routed through it).

This problem, known as the Ring Loading Problem, is $\mathcal{NP}$-complete in the general case [26]. It has been polynomialy solved whenever integer demand splitting is allowed [25, 26]. In the case where demand splitting is forbidden, Schrijver et al. [26] design a polynomial algorithm with the guarantee of exceeding the optimum by at most an additive term of 3/2 of the maximum demand. Their approach has been further developed by Khanna [16], leading to a polynomial $(1+\epsilon)$-approximation scheme.

The Ring Loading Problem is a multibin problem. The m edges of the ring are the m bins, and the n demands are the n objects (the height of an object is the demand). An object, i.e., a demand $d_{i,j}$, must be assigned to the set of edges between i and j (clockwise), or to the complementary set. This is a special case of $B|\text{set}_j|H_{\max}$.

1.2.2. *Document Analysis*

A group of experts has to examine documents, and each document must be examined by several experts according to its importance. A certain time is allocated for expertizing each of the documents. The experts correspond to the bins, the documents correspond to the objects. The objective is to expertize all documents as fast as possible. If every expert is competent for every document then it is a $Bm|\text{size}_j|H_{\max}$ problem; otherwise it is a $Bm|\text{set}_j|H_{\max}$ problem.

1.2.3. *File Backups*

A set of files (the objects) of given size (the height of the object) has to be saved on several disks (the bins). For security reasons, multiple copies must be made

(the width of the object), depending on the importance of the file. The objective is to minimize the number of disks of a given capacity. This is clearly a $B|\text{size}_j|M$ problem.

1.2.4. *Drug Testing*

As an example to introduce matchings, Lovász and Plummer [23] consider a pharmaceutical firm that wants to test n antibiotics on n volunteer subjects, with consideration of allergy cases.

Before a drug can be released on the market, however, it must be tested on several volunteers. A more realistic model would thus be to have n drugs to test on m volunteers, and each drug must be given to a certain number of these volunteers. Since the aim is to cure people and not to get them sick, no volunteer should be given too many drugs.

This is typically a $Bm|\text{set}_j|H_{\max}$ problem, where the bins are the m volunteers and the drugs are the n objects. The width of an object is the number of volunteers the drug must be given to, and its height is a measure of the dangerousness of the drug. The set_j field represents allergy compatibility.

1.2.5. *Software Development*

Several libraries must be written for a software. Each library is made of independant programs and hence the different programmers working on the same library do not have to synchronize their work. A library may be written by any number of programmers, and its processing time depends on this number.

Several objectives may be considered. If we have m programmers and we want to complete the entire project as soon as possible, it is a $Bm|\text{any}_j|H_{\max}$ problem. If we have a deadline (a bin's capacity) and we want to minimize the number of programmers, then we have a $B|\text{any}_j|M$ problem.

1.2.6. *Summer School Planning*

For a summer school, several classes must be chosen and programmed during the week. A class requires size_j times h_j hours, and two lessons of a same class must not occur on the same day. We want to organize as many classes as possible.

In this problem, the classes are the objects, the days are the bins (M days of H hours). It is a $B|\text{size}_j|N$ problem, or a $B|\text{size}_j|W$ problem if some classes are more important than others.

2. Packing *vs.* Scheduling

In this section, we recall some transformations to turn a schedule into a packing, and conversely, and their effect on the respective values $C_{\max}$ and $H_{\max}$. For proofs and details, the reader is referred to [20, 21].

2.1. DROPPING SCHEDULES

Consider a multiprocessor problem and one of its solutions represented on a Gantt chart. Turn this chart by 90°, and let the tasks drop: the result is a feasible solution of the associated multibin problem, where a task T_j of length p_j becomes an object O_j of height h_j (remember Figure 1). In other words, from any multiprocessor solution, one gets a multibin solution for the associated problem by removing idle time. By this dropping, the different parts of a same object may end up being packed at different levels.

Formally, let I be an instance of a multiprocessor problem and let S be a feasible schedule for I. We define the *dropping of* S, denoted by $S^{\downarrow}$, as the packing having the same allocations as S, i.e., O_j is packed into B_i in $S^{\downarrow}$ if and only if T_j is processed on P_i in S.

Notice that a dropping is always a feasible solution for the instance viewed as a multibin instance. Moreover, by this dropping operation, a multibin problem is a relaxation of its associated multiprocessor problem. However, even if the schedule is optimal, once idle times are removed, it may not be an optimal packing.

Hence it is very natural to evaluate how good is a packing obtained by relaxing an optimal schedule. We give an answer to this question in the size_j case with uniform processing times ($p_j = 1$), i.e., the problem $Pm|\text{size}_j, p_j = 1|C_{\max}$ (solved in time $\mathcal{O}(n)$ in [4]).

PROPOSITION 1. *Let $H^*_{\max}$ be the height of an optimal packing for an instance I, and let $H^{\downarrow}_{\max}$ be the value of the dropping of an optimal schedule. We have*:

$$H^{\downarrow}_{\max} \leq \frac{2m}{m+1} H^*_{\max} + \frac{m-1}{m+1}$$

and this bound is asymptotically tight.

A proof of this result is available in [21].

Figure 2 shows an example where the bound is tight.

2.2. LIFTING PACKINGS

Consider now the converse problem: we have a packing and we want to transform it into the best possible schedule.

Formally, let I be an instance of a multibin problem and let S be a feasible packing for this instance. A *lifting* of S is any schedule $S^{\uparrow}$, feasible for I as a multiprocessor instance, whose dropping is S. Moreover, a lifting $S^{\uparrow *}$ is said to be *optimal* if it is optimal among all liftings of S.

One may remark that lifting problems are exactly fix_j multiprocessor scheduling problems: in both cases, given the allocations of the tasks to the processors,

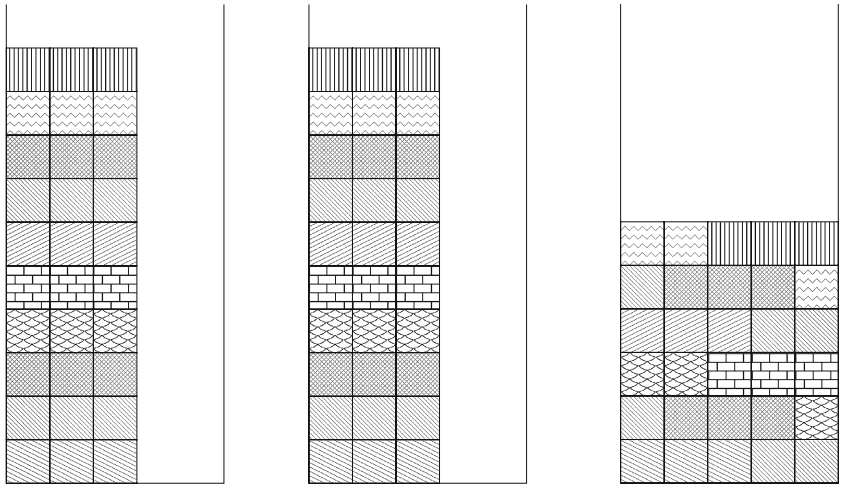

Figure 2. An example of a dropping with tight bound in Proposition 1. (a) An optimal schedule; (b) its dropping; (c) an optimal packing.

we are searching for the earliest completion time schedule. Hence, from the $\mathcal{NP}$ completeness of $P3|\,\mathrm{fix}_j|C_{\max}$ and $P|\,\mathrm{fix}_j, p_j = 1|C_{\max}$ follows:

PROPOSITION 2. *Finding an optimal lifting is $\mathcal{NP}$-hard, even for 3 bins or for equal-height objects*.

For two bins, the problem $B2|\mathrm{size}_j|H_{\max}$ is equivalent to $P2|\mathrm{size}_j|C_{\max}$, and more generally $Bm|\mathrm{size}_j \in \{1, m\}|H_{\max}$ is equivalent to $Pm|\mathrm{size}_j \in \{1, m\}|C_{\max}$. In those cases, lifting is polynomial (it is just a re-ordering of the objects). However, finding an optimal packing or scheduling is $\mathcal{NP}$-hard (but pseudo-polynomial), since this reduces to the problem $P2||C_{\max}$.

3. Complexity of Multibin Problems

Multibin problems include or extend several well-known $\mathcal{NP}$-complete problems. In this section, the complexity of those problems is studied, and some polynomial, or pseudo-polynomial, algorithms are given.

3.1. SOME $\mathcal{NP}$-COMPLETE CASES

Consider the general size_j problem, formulated as a decision problem as follows (the notation $[n]$ represents the set $\{1, 2, \ldots, n\}$):

PROBLEM 3 $[B|\mathrm{size}_j|\cdot]$.

Instance: a set $\mathbf{O}$ of n objects: $\{O_j = (h_j, \text{size}_j, w_j), j \in [n]\} \subset I\!N^{3n}$; a bound $H \in I\!N$, a bound $M \in I\!N$ and a bound $W \in I\!N$.
Question: is there a subset $\mathbf{O}'$ of the objects ($\mathbf{O}' = \{O'_j\} \subset \mathbf{O}$) with total weight exceeding W ($\sum_{O'_j \in \mathbf{O}'} w_j \geq W$) and such that there is a placement of $\mathbf{O}'$ into M bins $(B_1, \ldots, B_M)$ of height H ($\sum_{O'_j \in B_i} h_j \leq H, \forall i \in [M]$) satisfying the cardinality constraints ($card(\{B_i; O'_j \in B_i\}) = \text{size}_j, \forall j \in [n]$)?

This formulation has size $\mathcal{O}(n(\ln H + \ln M + \ln W))$. Note that a solution (the B_i) has size $\mathcal{O}(n \cdot M)$, which is not polynomially bounded by the instance size if the number of bins M is not fixed (M, as H, cannot be bounded since an object may have arbitrary big height and width)! Indeed, in this case, we do not know if the problem is in $\mathcal{NP}$; however, a solution is clearly a polynomial certificate for a "yes" answer if M is fixed and thus $Bm|\text{size}_j|\cdot$ is in $\mathcal{NP}$.

Several subcases are already well-known. $B2|\text{size}_j|H_{\max}$ corresponds to the partition problem; $Bm|\text{size}_j = 1|H_{\max}$ is the parallel scheduling problem $Pm||C_{\max}$; and $B^1|\text{size}_j|W$ is the knapsack problem (all $\mathcal{NP}$-complete and solvable in pseudo-polynomial time [13]). $B|\text{size}_j = 1|H_{\max}$ is the parallel scheduling problem $P||C_{\max}$, and $B|\text{size}_j = 1|M$ is the bin packing problem (both strongly $\mathcal{NP}$-complete [13]).

PROPOSITION 4. *$Bm|size_j|\cdot$ is $\mathcal{NP}$-complete*; *$B|\text{size}_j|\cdot$ is at least strongly $\mathcal{NP}$-complete*.

Proof. $Bm|\text{size}_j|\cdot$ reduces to the $Pm||C_{\max}$ problem by allowing only instances with: $W = n$, $H = C_{\max}$ and $\forall j \in [n] : \text{size}_j = 1, w_j = 1$. Hence it is $\mathcal{NP}$-complete.

Consider an instance of 3-partition [13]: $3m$ items of length l_j, and a bound B, with the question: is it possible to pack the items into m bins such that the sum of the lengths of the objects in any bin is B. Now consider the transformation of this 3-partition instance to a $B|\text{size}_j|\cdot$ instance: $M = m$, $H = B$, $W = n$, $n = 3m$ and $\forall j \in [n] : h_j = l_j, \text{size}_j = 1, w_j = 1$. This is a pseudo-polynomial transformation (as defined in [13], p. 101). As 3-partition is strongly $\mathcal{NP}$-complete, so is $B|\text{size}_j|\cdot$. □

Note that the above formulation is general for any of the $H_{\max}$, M, N and W criteria. For N, the w_j should be dropped, and for M and $H_{\max}$ both the w_j and the bound W should be dropped, thus decreasing the size of an instance. However the above results still hold. Note that this is not always the case: $B^1|\text{size}_j|W$ is the knapsack problem ($\mathcal{NP}$-complete [13]), whereas with all other objectives, the problem is trivial.

For the any_j case, one must specify the heights of an object, depending on its size. Therefore, a vector of size m must be given instead of the couple (h_j, size_j). The problem becomes:

PROBLEM 5 ($B|\text{any}_j|\cdot$).

Instance: a bound $M \in I\!N$, a set $\mathbf{O}$ of n objects: $\{O_j = ([h_j(s), s \in [M]], w_j), j \in [n]\} \subset I\!N^{(M+1)n}$; a bound $H \in I\!N$, and a bound $W \in I\!N$.
Question: is there a subset $\mathbf{O}'$ of the objects ($\mathbf{O}' = \{O'_j\} \subset \mathbf{O}$) with total weight exceeding W ($\sum_{O'_j \in \mathbf{O}'} w_j \geq W$) and such that there is a placement of $\mathbf{O}'$ into M bins $(B_1, \ldots, B_M)$ of height H ($\sum_{O'_j \in B_i} h_j(s_j) \leq H, \forall i \in [M]$, with $s_j = card(\{B_i; O'_j \in B_i\}), \forall j \in [n]$)?

The size of this problem is $\mathcal{O}(n(M \ln H + \ln W))$. The size of a solution is clearly polynomially bounded by this quantity: problems $Bm|\text{any}_j|\cdot$ and $B|\text{any}_j|\cdot$ are in $\mathcal{NP}$.

PROPOSITION 6. *$Bm|\text{any}_j|\cdot$ is $\mathcal{NP}$-complete; $B|\text{any}_j|\cdot$ is strongly $\mathcal{NP}$-complete.*

Proof. Consider an instance I of $B|\text{size}_j|\cdot$; it can be turned into an instance I' of $B|\text{any}_j|\cdot$ as follows: $h_j(s)$ is the height h_j if $s = \text{size}_j$, and $+\infty$ otherwise. In the process, the size of the instance increases from $\mathcal{O}(n(\ln H + \ln M + \ln W))$ to $\mathcal{O}(n(M \ln H + \ln W))$ (this transformation is not polynomial).

If $B|\text{any}_j|\cdot$ is not strongly $\mathcal{NP}$-complete, then there exists a pseudo-polynomial time algorithm that solves it, and hence there exists an algorithm that solves $B|\text{size}_j|\cdot$ in time polynomial in n, H, M, W. The strong $\mathcal{NP}$-completeness of $B|\text{size}_j|\cdot$ rules this out: and hence $B|\text{any}_j|\cdot$ is strongly $\mathcal{NP}$-complete.

Similarly, a transformation of an instance of $Bm|\text{size}_j|\cdot$ to an instance of $Bm|\text{any}_j|\cdot$ is polynomial (M is fixed). Thus, if $Bm|\text{any}_j|\cdot$ was polynomially solvable, that would contradict the $\mathcal{NP}$-completeness of $Bm|\text{size}_j|\cdot$. □

As for size_j problems, if we focus on particular objectives such as $H_{\max}$, M, or N, the size of an instance decreases, but the results still hold, for similar reasons.

For the set_j case, the complexity highly depends on the type of the sets. If any set is possible, then we must precise, for every set and every object, if the object fits in the set or not: hence we need a characteristic vector for the 2^M possible sets, for every object, and the size of an instance explodes to $\mathcal{O}(n(2^M + \ln H + \ln W))$! If M is fixed, the size becomes $\mathcal{O}(n(\ln H + \ln W))$ and hence the $Pm||C_{\max}$ problem is a polynomial restriction. Therefore, $Bm|\text{set}_j|\cdot$ is $\mathcal{NP}$-complete. The case $B|\text{set}_j|\cdot$ with only 2 complementary sets per object is also $\mathcal{NP}$-hard (the ring loading problem [26] is a particular case).

3.2. SOME POLYNOMIAL CASES

An approach similar to McNaughton's algorithm for $P|pmtn|C_{\max}$ [7, 24] solves the cases of objects of fixed size and unitary height, for any one of the $H_{\max}$, M or N objectives, in polynomial time (this was already known for $H_{\max}$ [21] and M [1]).

For all three cases, the key is that one knows in advance what space is needed to pack optimally the objects. Then we consider special solutions where all parts of an object are put into adjacent bins. That is: for every O_j there is an index i_j such that O_j is put in $B_{i_j}, B_{i_j+1}, \ldots, B_{i_j+\text{size}_j-1}$ (indices must be understood modulo m, that is: B_{i+m} is B_i). With regard to such solutions, we only need to give for every object O_j the index i_j to define completely a solution.

To build such solutions, we use a simple wrap-around algorithm (Algorithm 1) that puts O_1 in the first size_1 bins, then O_2 in the size_2 next bins and so on. At the end of the procedure the size_j parts of object O_j are packed into size_j different bins (hence the solution is feasible) and the difference of height between two bins is at most 1, since objects are all 1-unit high.

ALGORITHM 1. Wrap-Around ($m \in I\!N$)

$i = 1$
for $j = 1$ to n do
  $i_j = i$
  $i = ((i + \text{size}_j - 1) \bmod m) + 1$

This algorithm builds optimal solutions for the $H_{\max}$ and M criteria:

PROPOSITION 7. *$B|\text{size}_j, h_j = 1|H_{\max}$ is solvable in time $\mathcal{O}(n)$. Its optimal value is*:

$$H^*_{\max} = \left\lceil \frac{\sum_{j=1}^{n} \text{size}_j}{m} \right\rceil.$$

Proof. The given value is clearly a lower bound, and it is reached by a solution built by Wrap-Around[M^*], since the difference of height between two bins is at most one. Thus, it is the optimal value, and Algorithm 1 is optimal for $B|\text{size}_j, h_j = 1|H_{\max}$. □

PROPOSITION 8. *$B|\text{size}_j, h_j = 1|M$ is solvable in time $\mathcal{O}(n)$. Its optimal value is*:

$$M^* = \max\left\{ \max_j\{\text{size}_j\} \; ; \; \left\lceil \frac{\sum_{j=1}^{n} \text{size}_j}{H} \right\rceil \right\}.$$

Proof. The given value is obviously a lower bound. Actually Wrap-Around[M^*] builds a feasible solution with this number of bins. Hence, it is the optimal value, and Algorithm 1 is optimal for $B|\text{size}_j, h_j = 1|M$. □

The case N is slightly different, since we cannot *a priori* pack all objects. Therefore we use an adapted version (Algorithm 2), that builds optimal solutions for the N criterion.

ALGORITHM 2. Wrap-Around 2($m \in I\!N, H \in I\!N$)

(1) sort objects so that: $\text{size}_1 \leq \text{size}_2 \leq \ldots \leq \text{size}_n$
(2) $i = 1$, $V = mH$
    for $j = 1$ to n do
        if $\text{size}_j \leq V$ then
            $i_j = i$
            $i = ((i + \text{size}_j - 1) \bmod m) + 1$
            $V = V - \text{size}_j$
        else STOP

PROPOSITION 9. *$B|\text{size}_j, h_j = 1|N$ is solvable in time $\mathcal{O}(n \ln n)$. Its optimal value is*:

$$N^* = \arg\max\left\{k \leq n \Big| \sum_{j=1}^{k} \text{size}_j \leq mH; \text{size}_1 \leq \text{size}_2 \leq \ldots \leq \text{size}_n\right\}.$$

Proof. The given value is obviously computed by the modified version of the Wrap-Around Algorithm (Algorithm 2), in time $\mathcal{O}(n \ln(n))$ (sorting objects in step 1 dominates packing them in step 2). Furthermore, the solution is optimal: as the smallest objects are packed first, one cannot replace some packed objects by more objects left unpacked. □

Notice that the forth objective, W, is $\mathcal{NP}$-hard even in this size_j, $h_j = 1$ case and with the additional restriction that $H = 1$ since it is the knapsack problem.

When applied to more general cases (e.g., $h_j \geq 1$), the above algorithms do not provide optimal solutions; however, some performances may still be guaranteed. For instance, Algorithm 1 is a 2-approximation for the problem $B|\text{size}_j|H_{\max}$ if the objects are first sorted by non-increasing heights [19]. More generally, similar principles and other simple packing rules lead to very efficient approximation algorithms [19, 22].

3.3. SOME PSEUDO-POLYNOMIAL CASES

We now present a dynamic programming appoach to solve the any_j case in pseudo-polynomial time for a fixed m and the $H_{\max}$ criterion. When m is 2 or 3, similar results exist for multiprocessor-task scheduling [6, 12] and the pseudo-polynomial algorithm developed here is essentially the same as in [12]. However

the approach in [12] is based on the existence of particular "canonical" schedules and does not extend to scheduling problems with larger m (to be precise, the case $P4|\text{any}|C_{\max}$ is still open, but $P5|\text{any}|C_{\max}$ is strongly $\mathcal{N}\rightarrow\mathcal{P}$-hard). For the packing case, since there is no order of the objects within the bins, every packing is "canonical," and the dynamic programming approach works for any fixed m.

LEMMA 10. *The problem* $B2|\text{any}_j|H_{\max}$ *can be solved up to optimality in pseudo-polynomial time* $\mathcal{O}(nA^2)$, *where* $A = \sum_{j=1}^{n} \max(h_j(1), h_j(2))$.

Proof. For $j \in [N]$, let $P(j, H_1, H_2)$ be the best value (i.e., the minimum greatest height) for packing objects $j, j+1, \ldots, n$, knowing that the two bins have been filled up respectively to H_1 and H_2 by the packing of the $j-1$ first objects. An object O_j can be packed into B_1 only, into B_2 only, or into both bins. We have the following relations for all heights H_1 and H_2 in $[A]$:

$$\begin{aligned} j \in [N]: \quad P(j, H_1, H_2) &= \min\{P(j+1, H_1 + h_j(1), H_2), \\ &\qquad P(j+1, H_1, H_2 + h_j(1)), \\ &\qquad P(j+1, H_1 + h_j(2), H_2 + h_j(2))\} \\ j = n+1: \quad P(j, H_1, H_2) &= \max\{H_1, H_2\} \end{aligned}$$

The problem is to compute the value $P(1, 0, 0)$.

The array P is of size $\mathcal{O}(nA^2)$, where A is an upper bound on the maximal height of a solution. $A = \sum_{j=1}^{n} \max(h_j(1), h_j(2))$ is such a bound. Hence, solving $B2|\text{any}_j|H_{\max}$ to optimality requires no more than computing the array P, and it can be done in time $\mathcal{O}(nA^2)$ if we avoid to compute the same value twice. □

REMARK 1. This algorithm also works if the height of an object depends on the bins an object is packed into.

These results and algorithms can be extended to problems with more than two bins:

THEOREM 11. *For any fixed integer* m, *the problem* $Bm|\text{any}_j|H_{\max}$ *can be solved in pseudo-polynomial time* $\mathcal{O}(nA^m)$, *where* $A = \sum_{j=1}^{n} \max_{k=1..m} h_j(k)$.

Proof. The same principles as for Lemma 10 are used. We now define $P(j, H_1, H_2, \ldots, H_m)$ and we have the following relations for all heights $H_i \in [A]$:

$$\begin{aligned} j \in [n]: \quad P(j, H_1, \ldots, H_m) &= \min_K P(j+1, H_1 + h_j(|K|)\delta_{1K}, \ldots \\ &\qquad \ldots, H_m + h_j(|K|)\delta_{mK}) \\ j = n+1: \quad P(j, H_1, \ldots, H_m) &= \max_{i \in [m]} H_i \end{aligned}$$

where K is any non-empty set of bins, $|K|$ is the cardinality of K, and δ_{iK} is 1 if $B_i \in K$, 0 otherwise.

The purpose is again to compute $P(1,0,\ldots,0)$. This can be done iteratively and leads to a $\mathcal{O}(nA^m)$ algorithm, as each iteration is of constant time for a fixed m (finding the minimum of a fixed number of integers), and thus the algorithm runs in time $\mathcal{O}(nA^m)$. □

The above algorithm considers every possible set of bins, but it can also be used for particular sets, e.g., sets of fixed size (the size_j case) or given sets (the set_j case). Moreover if the objects are of equal height, the algorithm is polynomial for every fixed m. Therefore we have the following property:

COROLLARY 12. *For any fixed integer m, the problems $Bm|\text{size}_j|H_{\max}$ and $Bm|\text{set}_j|H_{\max}$ can be solved in pseudo-polynomial time and the problem $Bm|\text{set}_j, h_j = 1|H_{\max}$ can be solved in polynomial time.*

Further details and complements may be found in [20].

The dynamic programming approach works also for the W criterion:

THEOREM 13. *For any fixed integer m, the problem $Bm|\text{any}_j|W$ can be solved in pseudo-polynomial time $\mathcal{O}(nH^m)$.*

Proof. Let us define $Q(j,\bar{H}_1,\bar{H}_2,\ldots,\bar{H}_m)$ as the maximal weighted number of objects of $\{O_j,O_{j+1},\ldots,O_n\}$ that we can pack, knowing that the remaining height in bin B_i is $\bar{H}_i$. We want to compute $Q(1,H,\ldots,H)$ and we have the following relations:

$$\begin{aligned} j\in[n]: \quad Q(j,\bar{H}_1,\ldots,\bar{H}_m) \;&= \\ &\max_K \delta_K w_j + Q(j+1,\bar{H}_1 - h_j(|K|)\delta_{1K},\ldots \\ &\ldots,\bar{H}_m - h_j(|K|)\delta_{mK}) \\ j=n+1: \quad Q(j,\bar{H}_1,\ldots,\bar{H}_m) \;&= 0 \end{aligned}$$

where K is any subset of $[m]$ where O_j fits (that is such that: $\forall B_i\in K: \bar{H}_i \geq h_j(|K|)$), $|K|$ is its cardinality; δ_{iK} is 1 if $B_i\in K$ and 0 otherwise; δ_K is 0 if K is empty, 1 otherwise; and w_j is the weight of object O_j. This relation simply means that we can either not pack O_j (this corresponds to $K=\emptyset$, and $\delta_K=0$), either pack it in any set of bins it fits in.

Computing $Q(1,H,\ldots,H)$ can be done iteratively and leads to a $\mathcal{O}(nH^m)$ algorithm, as each iteration is of constant time for a fixed m (finding the maximum of a fixed number of integers). □

As for the $H_{\max}$ criterion, the algorithm can also be used for particular sets, e.g., sets of fixed size (the size_j case) or given sets (the set_j case), or for a constant weight function (the N criterion). Therefore we have the following:

COROLLARY 14. *For any fixed integer m, the problems $Bm|\text{size}_j|N$ and $Bm|\text{size}_j|W$ and $Bm|\text{set}_j|N$ and $Bm|\text{set}_j|W$, $Bm|\text{any}_j|N$ can be solved in pseudo-polynomial time.*

Table I. Complexity of multibin problems.

$B\|\text{size}_j,\ h_j=1\|H_{\max}$	$\mathcal{O}(n)$	Proposition 7
$B2\|\text{size}_j\|H_{\max}$	$\mathcal{NP}$-h, p$\mathcal{P}$	Partition
$Bm\|\text{size}_j=1\|H_{\max}$	$\mathcal{NP}$-h, p$\mathcal{P}$	$Pm\|\|C_{\max}$
$B\|\text{size}_j=1\|H_{\max}$	s$\mathcal{NP}$-h	$P\|\|C_{\max}$
$Bm\|\text{size}_j\|H_{\max}$	$\mathcal{NP}$-h, p$\mathcal{P}$	Proposition 4, Corollary 12
$B\|\text{size}_j\|H_{\max}$	at least s$\mathcal{NP}$-h	Proposition 4
$Bm/\text{set}_j,\ h_j=1\|H_{\max}$	Polynomial	Corollary 12
$Bm\|\text{set}_j\|H_{\max}$	$\mathcal{NP}$-h, p$\mathcal{P}$	Corollary 12
$Bm\|\text{any}_j\|H_{\max}$	$\mathcal{NP}$-h, p$\mathcal{P}$	Proposition 4, Theorem 11
$B\|\text{any}_j\|H_{\max}$	s$\mathcal{NP}$-h	Proposition 6
$B\|\text{size}_j,\ h_j=1\|M$	$\mathcal{O}(n)$	Proposition 8
$B\|\text{size}_j=1\|M$	s$\mathcal{NP}$-h	Bin packing
$B\|\text{size}_j\|M$	at least s$\mathcal{NP}$-h	Proposition 4
$B\|\text{set}_j\|M$	s$\mathcal{NP}$-h	Proposition 6
$B\|\text{size}_j,\ h_j=1\|N$	$\mathcal{O}(n\ln(n))$	Proposition 9
$Bm\|\text{size}_j\|N$	$\mathcal{NP}$-h, p$\mathcal{P}$	Proposition 4, Corollary 14
$B\|\text{size}_j\|N$	at least s$\mathcal{NP}$-h	Proposition 4
$Bm\|\text{set}_j\|N$	$\mathcal{NP}$-h, p$\mathcal{P}$	Corollary 14
$Bm\|\text{any}_j\|N$	$\mathcal{NP}$-h, p$\mathcal{P}$	Proposition 6, Corollary 14
$B\|\text{any}_j\|N$	s$\mathcal{NP}$-h	Proposition 6
$B^1\|\text{size}_j,\ h_j=1\|W$	$\mathcal{NP}$-h, p$\mathcal{P}$	Knapsack
$Bm\|\text{size}_j\|W$	$\mathcal{NP}$-h, p$\mathcal{P}$	Proposition 4, Corollary 14
$Bm\|\text{set}_j\|W$	$\mathcal{NP}$-h, p$\mathcal{P}$	Corollary 14
$B\|\text{size}_j\|W$	at least s$\mathcal{NP}$-h	Proposition 4
$Bm\|\text{any}_j\|W$	$\mathcal{NP}$-h, p$\mathcal{P}$	Proposition 6, Theorem 13
$B\|\text{any}_j\|W$	s$\mathcal{NP}$-h	Proposition 6

3.4. SUMMARY OF COMPLEXITY RESULTS

Table 1 sums up the above discussions with the complexity results for the different objectives ($\mathcal{NP}$-h: $\mathcal{NP}$-hard; s$\mathcal{NP}$-h: strongly $\mathcal{NP}$-hard; p$\mathcal{P}$: pseudo-polynomial).

4. A FPTAS for $B_m|\text{any}_j|H_{\max}$

In this section we describe a fully polynomial time approximation scheme (FPTAS) for the problem $Bm|\text{any}_j|H_{\max}$, that is an algorithm which is a $(1+\epsilon)$-approximation and which runs in time polynomial both in the instance size and in $1/\epsilon$, for every fixed m.

The technique used in this section, known as the *scaling technique* (or *fixed partitioning technique*), is a very common one, used successfully on various problems that admit pseudo-polynomial algorithms [2].

We know from Theorem 11 that there exists a pseudo-polynomial time algorithm *Alg* that optimally solves $Bm|\text{any}_j|H_{\max}$ in time $\mathcal{O}(nA^m)$, where $A=\sum_{j=1}^{n}\max_{k=1..m}h_j(k)$.

Algorithm *Alg* may then be used to solve approximately $Bm|\text{any}_j|H_{\max}$. We first re-scale the heights of the objects by a factor δ: $h_j \to h'_j = \left\lfloor \frac{h_j}{\delta} \right\rfloor$. Hence, the computational time is also divided by δ, when the optimal solution of the re-scaled instance is computed. To get a feasible solution for the initial instance, we just need to reconvert the objects to the initial heights. In this process, the additive error is bounded by $n\delta$. If δ is suitably chosen, we obtain a FPTAS, as done in Algorithm 3 and proved below.

In what follows, $h_{\max}$ is the maximal possible height for an object, i.e.: $h_{\max} = \max_{j,s} h_j(s)$.

ALGORITHM 3. FPTAS (instance I, positive real ϵ)

(1) compute $\delta = \epsilon \frac{h_{\max}}{n}$.
(2) re-scale I to get I': $h'_j = \left\lfloor \frac{h_j}{\delta} \right\rfloor$.
(3) solve I': $S' = Alg(I')$.
(4) return S, similar to S' but with the initial heights.

THEOREM 15. *Algorithm* 3 *is a FPTAS for* $Bm|\text{any}_j|H_{\max}$; *hence also for* $Bm|\text{size}_j|H_{\max}$ *and* $Bm|\text{set}_j|H_{\max}$. *It runs in time* $\mathcal{O}(\frac{n^{2m+1}}{\epsilon^m})$, *which is polynomial in* n *and* $1/\epsilon$ *if* m *is fixed*.

Proof. For this proof, we need several intermediate instances and solutions. Only the heights of the objects differ from one to the other. In order of appearance, we have the following.

I' is the instance with heights $\left\lfloor \frac{h_j}{\delta} \right\rfloor$; S' is an optimal solution for it.

S'' is S' with the heights multiplied by δ. That is: S'' is the solution for I'' with heights $\delta \left\lfloor \frac{h_j}{\delta} \right\rfloor$.

S is the final solution with initial heights h_j.

First, remark that, since S' is optimal for I', then S'' is optimal for I'', as we just multiply all heights by a constant factor. Moreover: $\forall \delta > 0 : \delta \left\lfloor \frac{h_j}{\delta} \right\rfloor \leq h_j$, and thus $H^{S''}_{\max} \leq H^*_{\max}$ (where $H^*_{\max}$ refers to the optimal value for I).

When getting back to the initial heights – that is when we convert S'' to S – the additive error introduced is bounded. Indeed, $\delta \left\lfloor \frac{h_j}{\delta} \right\rfloor + \delta \geq h_j$: the error on one object is at most δ. As there are at most n objects per bin, the additive total error is at most $n\delta$. Hence: $H^S_{\max} \leq H^*_{\max} + n\delta$.

Now, remark that $H^*_{\max} \geq h_{\max}$. Thus:

$$\frac{H^S_{\max}}{H^*_{\max}} \leq 1 + \frac{n\delta}{h_{\max}} = 1 + \epsilon$$

and Algorithm 3 is a $(1+\epsilon)$-approximation.

The overall running time is dominated by step 3. Theorem 11 insures that this step is done in time less than:

$$\mathcal{O}\left(n(\sum_{j=1}^{n} h'_j)^m\right) \leq \mathcal{O}\left(n(n\frac{h_{\max}}{\delta})^m\right) \leq \mathcal{O}\left(n(\frac{n^2}{\epsilon})^m\right) \leq \mathcal{O}\left(\frac{n^{2m+1}}{\epsilon^m}\right)$$

which is polynomial both in the instance size, if m is fixed, and in $\frac{1}{\epsilon}$. □

5. Conclusions

This paper updates the existing models for multibin problems. In particular the new criteria N and W are introduced and studied. Complexity results are also extended with a new pseudo-polynomial time algorithm for the W criterion, and a FPTAS for the $H_{\max}$ criterion.

Considering the results obtained in this study, the first two questions that arise are: is there a pseudo-polynomial time algorithm that solves $B|\text{size}_j, h_j = 1|W$ or is this problem strongly $\mathcal{NP}$-hard? and: is there a FPTAS for $Bm|\text{any}_j|W$?

From a practical point of view, the above methods cannot be used as they are. The algorithms for objects of height 1 are very efficient, but these cases are too simple in general. On the contrary, the pseudo-polynomial algorithms and the FPTAS are very general, but would turn out to be too slow, even though polynomial, as soon as the number of bins m exceeds 4 or 5. However, they point out several promising principles and give a first and precise insight of what can be expected from them. In particular, the fast algorithms of Section 3.2 use special solutions where objects are packed into adjacent bins, which are especially well designed for fast heuristicxs. Such algorithms can be easily extended to more general cases, and guarantees on their performances should be obtained.

Furthermore, the results presented in this paper show that techniques from the well-studied fields of scheduling (for the $H_{\max}$ criterion) and bin-packing (for the M criterion) can be successfully extended to multibin packings. For instance, heuristic rules can be adapted and are likely to lead to efficient practical approximation algorithms, even though they would be more challenging to study.

References

1. Abouelaoualim, A.: Heuristiques avec performances pour les problèmes de rangements multiboîtes.' Master's thesis, INPG, Grenoble, France, 2003.
2. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A. and Protasi, M.: *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer, Berlin, 1999.

3. Baker, B. S. and Coffman Jr, E. G.: Mutual exclusion scheduling, *Theor. Comp. Sci.* **162**(2) (1996) 225–243.
4. Błażewicz, J., Drabowski, M. and Węglarz, J.: Scheduling multiprocessor tasks to minimize schedule length. *IEEE Trans. Comput.* **C-35** (1986) 389–393.
5. Błażewicz, J., Drozdowski, M. and Ecker, K.: Management of resources in parallel systems, in J. Błażewicz, K. Ecker, B. Plateau and D. Trystram (eds.), *Handbook on Parallel and Distributed Processing*, Springer, Heidelberg, 1999, pp. 216–293.
6. Błażewicz, J., Drozdowski, M. and Węglarz, J.: Scheduling multiprocessor tasks – a survey, *Int. J. Microcomput. Appl.* **13**(2) (1994) 89–97.
7. Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G. and Węglarz, J.: *Scheduling Computer and Manufacturing Processes*, Springer-Verlag, 1996.
8. Bodlaender, H. L., Jansen, K. and Woeginger, G.: Scheduling with incompatible jobs, *Discrete Appl. Math.* **55**(3) (1994) 219–232.
9. Chen, B., Potts, C. N. and Woeginger, G. J.: A review of machine scheduling: Complexity, algorithms and approximability, in D.-Z. Du and P. Pardalos (eds.), *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, Dordrecht, 1998, pp. 21–169.
10. Coffman Jr, E. G., Garey, M. R. and Johnson, D. S.: Approximation algorithms for bin-packing: a survey (Chap. 2), in D. S. Hochbaum (ed.), *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, Boston, 1997, pp. 46–93.
11. Drozdowski, M.: Scheduling multiprocessor tasks – An overview, *Eur. J. Oper. Res.* **94** (1996) 215–230.
12. Du, J. and Leung, J. Y.-T.: Complexity of scheduling parallel task systems, *SIAM J. Discrete Math.* **2**(4) (1989) 473–487.
13. Garey, M. R. and Johnson, D. S.: *Computers and Intractability* (*A Guide to the Theory of NP-Completeness*), Freeman, San Francisco, CA, 1979.
14. Irani, S. and Leung, V. J.: Scheduling with conflicts, and applications to traffic signal control, in *SODA: ACM-SIAM Symposium on Discrete Algorithms* (*A Conference on Theoretical and Experimental Analysis of Discrete Algorithms*), 1996, pp. 85–94.
15. Jansen, K.: An approximation scheme for bin packing with conflicts, *J. Comb. Optim.* **3** (1999) 363–377.
16. Khanna, S.: A polynomial time approximation scheme for the SONET ring loading problem, *Bell Lab. Tech. J.*, 1997, pp. 36–41.
17. Labbé, M., Laporte, G. and Martello, S.: An exact algorithm for the dual bin packing problem, *Oper. Res. Lett.* **17**(1) (1995) 9–18.
18. Labbé, M., Laporte, G. and Martello, S.: Upper bounds and algorithms for the maximum cardinality bin packing problem, *Eur. J. Oper. Res.* **149**(3) (2003) 490–498.
19. Lemaire, P.: Rangements d'objets multiboîtes : modèles et algorithmes,' Ph.D. thesis, Université Joseph Fourier (Grenoble 1), Grenoble, France, 2004.
20. Lemaire, P., Finke, G. and Brauner, N.: Packing of Multibin Objects, Technical Report 69, Les Cahiers du Laboratoire Leibniz-IMAG, http://wwwleibniz.IMAG.fr/LesCahiers, 2002.
21. Lemaire, P., Finke, G. and Brauner, N.: Packing of multibin objects, in *IEPM'03, International Conference on Industrial Engineering and Production Management*, Vol. 1. Porto, Portugal, 2003, pp. 422–431. available update (www-leibniz.imag.fr/~lemaire).
22. Lemaire, P., Finke, G. and Brauner, N.: The best-fit rule for multibin packing: An extension of Graham's list algorithms, in G. Kendall, E. Burke, S. Petrovic and M. Gendreau (eds.), *Multidisciplinary Scheduling; Theory and Applications*, Springer, Heidelberg, 2005, pp. 269–286.
23. Lovász, L. and Plummer, M. D.: *Matching Theory*, Akadémiai Kiadó, Budapest, 1986.
24. McNaughton, R.: Scheduling with deadlines and loss functions, *Manage. Sci.* **6** (1959) 1–12.

25. Myung, Y.-S.: An efficient algorithm for the ring loading problem with integer demand splitting, *SIAM J. Discrete Math.* **14**(3) (2001) 291–298.
26. Schrijver, A., Seymour, P. and Winkler, P.: The ring loading problem, *SIAM J. Discrete Math.* **11**(1) (1998) 1–14.
27. SONET, The SONET Home Page, http://www.sonet.com.
28. Wu, T.-H.: *Fiber Network Service Survivability*, Artech House, Norway, MA, 1992.