Math. Program., Ser. A 91: 11–31 (2001)

Digital Object Identifier (DOI) 10.1007/s101070100243

Sándor P. Fekete · Jörg Schepers

New classes of fast lower bounds for bin packing problems*

Received: August 11, 1998 / Accepted: February 1, 2001
Published online September 17, 2001 – © Springer-Verlag 2001

Abstract. The bin packing problem is one of the classical NP-hard optimization problems. In this paper, we present a simple generic approach for obtaining new fast lower bounds, based on dual feasible functions. Worst-case analysis as well as computational results show that one of our classes clearly outperforms the previous best "economical" lower bound for the bin packing problem by Martello and Toth, which can be understood as a special case. In particular, we prove an asymptotic worst-case performance of 3/4 for a bound that can be computed in linear time for items sorted by size. In addition, our approach provides a general framework for establishing new bounds.

Key words. bin packing – lower bounds – dual feasible functions

1. Introduction

The bin packing problem (BPP) can be described as follows: Given a set of n "items" with sizes $x_1, \ldots, x_n$, and a supply of identical "containers" of capacity C, decide how many containers are necessary to pack all the items. This task is one of the classical problems of combinatorial optimization and NP-hard in the strong sense–see Garey and Johnson [11]. An excellent survey by Coffmann, Garey, and Johnson can be found as Chapter 2 in the book [2].

Over the years, many clever methods have been devised to deal with the resulting difficulties. Most notably, Fernandez de la Vega and Lueker [10], and Karmarkar and Karp [15] have developed polynomial-time approximation schemes that allow it to approximate an optimal solution within $1 + \varepsilon$ in polynomial (even linear) time, for any fixed ε. However, these landmark results are mostly of theoretical value, due to the enormous size of the constants. In practice as well as in theoretical analysis, most commonly heuristics like FIRST FIT DECREASING (FFD) are used, sometimes in combination with local improvement heuristics. In order to get a fast estimate on the quality of such heuristic solutions, it is desirable to have equally fast lower bounds. A classic

S.P. Fekete: Abteilung Mathematische Optimierung, Technische Universität Braunschweig, 38106 Braunschweig, Germany, e-mail: sandor.fekete@tu-bs.de

J. Schepers: IBM Global Services, IBM Germany, 50968 Köln, Germany
e-mail: joerg_schepers@de.ibmmail.com

Mathematics Subject Classification (2000): 90C27

* A preliminary version of this paper appears in the Proceedings of the 6th International Integer Programming and Combinatorial Optimization Conference [5]. This work was supported by the German Federal Ministry of Education, Science, Research and Technology (BMBF, Förderkennzeichen 01 IR 411 C7) while both authors were employed at the Center for Parallel Computing (ZPR) of the Universität zu Köln, 50623 Köln, Germany

example of such a bound is the easy volume bound L_1. While this bound is very simple to compute, its quality is limited, and its relative performance can be as small as $1/2$ OPT. This has motivated Martello and Toth [18,19] to develop the fast bound L_2 that has a worst-case performance of $2/3$ OPT.

In this paper, we present a simple and fast generic approach for obtaining lower bounds, based on *dual feasible functions*. Assuming that the items are sorted by size, our bounds can be computed in linear time with small constants, a property they share with the lower bound L_2. One of our classes of lower bounds can be interpreted as a systematic generalization of L_2. Its worst-case performance turns out to be $(3/4\ \mathrm{OPT} - 1)$. As Chan, Simchi-Lavi, and Bramel showed in [3], 3/4 also happens to be the worst-case performance of the column generation bound, which is much more expensive to compute. (Note, however, that the *asymptotic* performance of the column generation bound is still an unsolved problem, since [3] only provides a specific example with performance $3/4$.)

It should be emphasized that our main motivation for developing and studying these new bounds has not been asymptotic worst-case performance. (Clearly, the approximation schemes mentioned above fare better in that category.) However, our bounds are extremely fast to compute, so they can be used quite easily in practice. This is illustrated by computational results, which indicate that our generalization yields a clear improvement in performance over previous "fast" bounds. This provides some evidence for their practical usefulness, e.g., in the context of a branch and bound framework. Other interesting scenarios arise when we want to complement a fast heuristic method like FFD by an equally fast lower bound, such that we may detect a small gap when the heuristic solution is close to optimal. Finally, our systematic approach is suited to simplify and improve other hand-tailored types of bounds.

The rest of this paper is organized as follows. In Sect. 2, we give an introduction to some "fast" lower bounds, in particular the context of the volume bound L_1, and the bound L_2 by Martello and Toth. In Sect. 3, we give an introduction to dual feasible functions and show how they provide a framework for obtaining fast and simple lower bounds for the BPP. In Sect. 4, we introduce a particular class of bounds that can be interpreted as a generalization of L_2, and we show that this improves the asymptotic worst-case performance from $2/3$ to $3/4$. In Sect. 5, we give computational results illustrating the usefulness of our bounds. Section 6 concludes with a discussion of possible extensions, including higher-dimensional packing problems.

2. Elementary bounds

In this section we give a basic introduction to some well-known bounds for the bin packing problem, and we discuss their performance. One particular performance measure that comes up in this context is the *worst-case performance*:

Definition 1 ((asymptotic) worst-case performance). *Let L be a lower bound for a minimization problem P. Let $OPT(I)$ denote the optimal value of P for an instance I. Then*

$$r(L) := \sup\left\{\frac{L(I)}{OPT(I)} \mid I \text{ is an instance of } P\right\} \tag{1}$$

is called the worst-case performance *and*

$$r_\infty(L) := \lim_{s \to \mathbb{R}_0^+} \sup \left\{ \frac{L(I)}{OPT(I)} \mid I \text{ is an instance of } P \text{ with } OPT(I) \geq s \right\} \tag{2}$$

is called the asymptotic worst-case performance *of L.*

2.1. The class L_1 and its relatives

In the following, we assume without loss of generality that the size C of the container is normalized to 1. Then one of the easiest lower bounds for the BPP is the total volume of all items, rounded up to the next integer. For a normalized BPP instance $I := (x_1, \ldots, x_n)$, this bound can be formulated as

$$L_1(I) := \left\lceil \sum_{i=1}^{n} x_i \right\rceil. \tag{3}$$

A standard approach for obtaining bounds for combinatorial optimization problem is to consider relaxations of an integer programming formulation. As described in [19], pp. 224 ff., the bound L_1 can also be obtained as the optimal solution of the LP-relaxation for the following ILP that describes the BPP:

$$\begin{aligned}
\min \quad & \textstyle\sum_{i=1}^{n} y_i && && (4)\\
\text{subject to } & \textstyle\sum_{j=1}^{n} x_j z_{ij} \leq y_i, && i = \{1, \ldots, n\}, && (5)\\
& \textstyle\sum_{i=1}^{n} z_{ij} = 1, && j = \{1, \ldots, n\}, && (6)\\
& y_i \in \{0, 1\}, && i = \{1, \ldots, n\}, && (7)\\
& z_{ij} \in \{0, 1\}, && i, j = \{1, \ldots, n\}. && (8)
\end{aligned}$$

(Here, $y_i = 1$ indicates that bin i is used, and $z_{ij} = 1$ indicates that item j is assigned to bin i.) In this formulation, L_1 can be interpreted as the solution over the continuous relaxation (i.e., $z_{ij} \in [0, 1]$ instead of 8) of the BPP, where items are allowed to be split and the pieces may be packed into several bins. It is also shown in [19] that the bound L_1 dominates the *surrogate relaxation*, which is given, for a positive vector (π_i) of multipliers, by

$$\begin{aligned}
\min \quad & \textstyle\sum_{i=1}^{n} y_i && && (9)\\
\text{subject to } & \textstyle\sum_{i=1}^{n} \pi_i \sum_{j=1}^{n} x_j z_{ij} \leq \pi_i y_i, \; i = \{1, \ldots, n\}, && && (10)\\
& (6), (7), (8), &&
\end{aligned}$$

and the *Lagrange relaxation*, which is given, for a positive vector (μ_i) of multipliers, by

$$\begin{aligned}
\min \; & \textstyle\sum_{i=1}^{n} y_i + \sum_{i=1}^{n} \mu_i \left(\sum_{j=1}^{n} x_j z_{ij} - y_i \right) && && (11)\\
\text{subject to } & (6), (7), (8), &&
\end{aligned}$$

According to their numerical experiments, L_1 approximates the optimal value very well, as long as there are sufficiently many items of small size, meaning that the remaining capacity of bins with big items can be exploited. If this is not the case, then the ratio between L_1 and the optimal value can reach a worst-case performance of $r(L_1) = \frac{1}{2}$, as shown by the class of examples with $2k$ items of size $\frac{1}{2} + \frac{1}{2k}$:

Observation 1 (Johnson [13]). $r(L_1) = \frac{1}{2}$.

2.2. The class L_2

In the situation which is critical for L_1 ("not enough small items to fill up bins"), it makes sense to concentrate on the "large" items. This yields the bound L_2, which was first introduced by Martello and Toth [18,19]: For a BPP instance I, let

$$L_2(I) := \max_{\epsilon \in [0, \frac{1}{2}]} (|\{x_i \in I \mid x > 1 - \epsilon\}| + L_1(\{x_i \in I \mid \epsilon \leq x_i \leq 1 - \epsilon\}) . \qquad (12)$$

The worst-case performance is improved significantly:

Theorem 1 (Martello and Toth). $r(L_2) = \frac{2}{3}$.

This bound of $\frac{2}{3}$ is tight: Consider the class of BPP instances with $6k$ items of size $\frac{1}{3} + \frac{1}{6k}$.

There is little hope that a lower bound for the BPP that can be computed in polynomial time can reach an absolute worst-case performance above $\frac{2}{3}$. Otherwise, the $\mathcal{NP}$-hard problem PARTITION (see [11], p. 47) of deciding whether two sets of items can be split into two sets of equal total size could be solved in polynomial time.

L_2 can be computed in time $O(n \log n)$. The computational effort is determined by sorting the items by their size, the rest can be performed in linear time:

Lemma 1 (Martello and Toth). *Consider a BPP instance $I := (x_1, \ldots, x_n)$. If the sizes x_i are pre-sorted by size, then $L_2(I)$ can be computed in time $O(n)$.*

The idea for evaluating the maximization over $\epsilon \in [0, \frac{1}{2}]$ efficiently is to keep track of the best objective value for $\epsilon \in [0, \delta]$ while δ increases from 0 to $1/2$. Clearly, we only have to examine values where δ or $1 - \delta$ coincide with an item size x_i. At any such "event", we only need to update items of this critical size, so no item will cause more than one update. Since these updates can be performed in $O(1)$ amortized time, we can evaluate L_2 in time $O(n)$ for pre-sorted items.

3. Dual feasible functions

Now we turn to the mathematical discussion of dual feasible functions, which will then be used for a general framework for describing lower bounds for the BPP. For the rest of this and the following section, we assume without loss of generality that the items have size $x_i \in [0, 1]$, and the container size C is normalized to 1. Then we introduce the following:

Definition 2 (Dual Feasible Functions). *A function* $u : [0, 1] \to [0, 1]$ *is called* dual feasible*, if for any finite set S of nonnegative real numbers, we have the relation*

$$\sum_{x \in S} x \leq 1 \implies \sum_{x \in S} u(x) \leq 1. \tag{13}$$

Dual feasible functions have been used in the performance analysis of heuristics for the BPP, first by Johnson [13], then by Lueker [16]; see Coffman and Lueker [4] for a more detailed description. The Term (which was first introduced by Lueker [16]) refers to the fact that for any dual feasible function u and for any bin packing instance with item sizes $x_1, \ldots, x_n$, the vector $(u(x_1), \ldots, u(x_n))$ is a feasible solution for the dual of the corresponding fractional bin packing problem (see [15]). By definition, convex combination and compositions of dual feasible functions are dual feasible.

We show in this paper that dual feasible functions can be used for improving lower bounds for the one-dimensional bin packing problem. This is based on the following easy lemma.

Lemma 2. *Let* $I := (x_1, \ldots, x_n)$ *be a BPP instance and let u be a dual feasible function. Then any lower bound for the transformed BPP instance* $u(I) := (u(x_1), \ldots, u(x_n))$ *is also a lower bound for I.*

By using a set of dual feasible functions $\mathcal{U}$ and considering the maximum value over the transformed instances $u(I),\ u \in \mathcal{U}$, we can try to obtain even better lower bounds.

In [16], a particular class of dual feasible functions is described; it relies on a special rounding technique. For a given $k \in \mathbb{N}$, consider the stair function $u^{(k)}$ that maps (for $i \in \{1, \ldots, k\}$) all values from the interval $[\frac{i}{k+1}, \frac{i+1}{k+1})$ onto the value $\frac{i}{k}$. 1 is mapped to 1. We give a slightly improved version and a simple proof that these functions are dual feasible.

Theorem 2. *Let* $k \in \mathbb{N}$. *Then*

$$u^{(k)} : [0, 1] \to [0, 1]$$
$$x \mapsto \begin{cases} x, & \text{for } x(k+1) \in \mathbb{Z} \\ \lfloor (k+1)x \rfloor \frac{1}{k}, & \text{else} \end{cases}$$

is a dual feasible function.

Proof. Let S be a finite set of nonnegative real numbers with $\sum_{x_i \in S} x_i \leq 1$. We have to show that $\sum_{x_i \in S} u^{(k)}(x_i) \leq 1$. Let $T := \{x_i \in S \mid x_i(k+1) \in \mathbb{Z}\}$. Clearly, we only need to consider the case $S \neq T$. Then

$$\begin{aligned}(k+1) \sum_{x_i \in T} u^{(k)}(x_i) + k \sum_{x_i \in S \setminus T} u^{(k)}(x_i) &= (k+1) \sum_{x_i \in T} x_i + \sum_{x_i \in S \setminus T} \lfloor (k+1)x_i \rfloor \\ &< (k+1) \sum_{x_i \in S} x_i.\end{aligned}$$

By definition of $u^{(k)}$, the terms $(k+1)\sum_{x\in T} u^{(k)}(x)$ and $k\sum_{x\in S\setminus T} u^{(k)}(x)$ are integer, so by virtue of $\sum_{x_i\in S} x_i \leq 1$, we have the inequality $(k+1)\sum_{x\in T} u^{(k)}(x) + k\sum_{x\in S\setminus T} u^{(k)}(x) \leq k$, implying $\sum_{x\in S} u^{(k)}(x) \leq 1$.

□

The rounding mechanism is visualized in Fig. 1, showing the difference of the stair functions $u^{(k)}$, $k \in \{1,\ldots,4\}$, and the identity *id* over the interval $[0,1]$. For the BPP, we mostly try to increase the sizes by a dual feasible function, since this allows us to obtain a larger, i.e., better bound. The hope is to find a $u^{(k)}$ for which as many items as possible are in the "win zones" – the subintervals of $[0,1]$ for which the difference is positive.

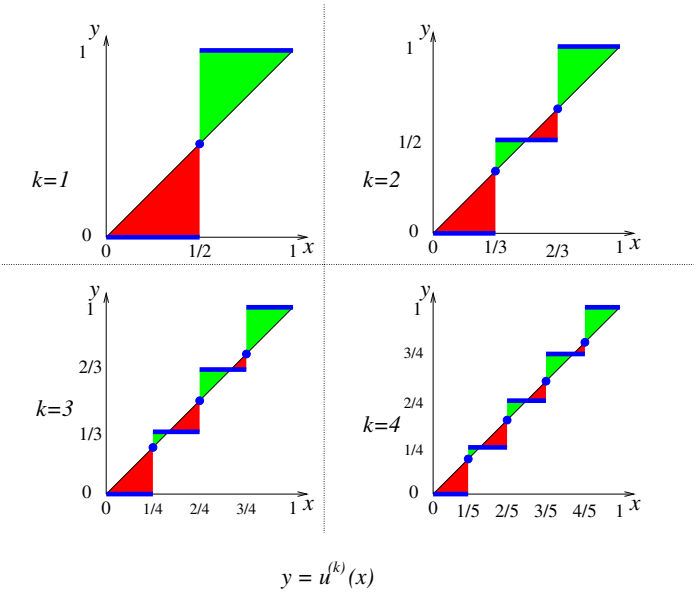

Fig. 1. Win and loss zones for $u^{(k)}$

The following class of dual feasible functions is the implicit basis for the bin packing bound L_2 by Martello and Toth [18,19]. This bound is obtained by neglecting all items smaller than a given value ϵ. We account for these savings by increasing all items of size larger than $1-\epsilon$. Figure 2 shows the corresponding win and loss zones.

Theorem 3. *Let* $\epsilon \in [0, \frac{1}{2}]$. *Then*

$$U^{(\epsilon)} : [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} 1, & \text{for } x > 1-\epsilon \\ x, & \text{for } \epsilon \leq x \leq 1-\epsilon \\ 0, & \text{for } x < \epsilon \end{cases}$$

is a dual feasible function.

Proof. See Fig. 2 for the win and loss zones. Let S be a finite set of nonnegative real numbers, with $\sum_{x\in S} x \leq 1$. We consider two cases. If S contains an element larger than $1-\epsilon$, then all other elements have to be smaller than ϵ. Hence we have $\sum_{x\in S} U^{(\epsilon)}(x) = 1$. If all elements of S have at most size $1-\epsilon$, we have $\sum_{x\in S} U^{(\epsilon)}(x) \leq \sum_{x\in S} x \leq 1$.

□

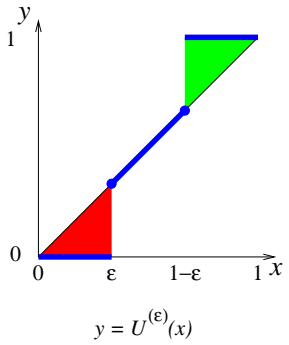

Fig. 2. Win and loss zones for $U^{(\epsilon)}$

We give a third example for a class of dual feasible functions to illustrate that their appearance may vary considerably. (This third class of dual feasible functions can be used in the context of higher-dimensional packing. It dominates and generalizes bounds that were hand-tailored for the two-dimensional and three-dimensional BPP by Martello and Vigo [20], and Martello, Pisinger, and Vigo [17]. Since those bounds are somewhat complicated to describe, the interested reader is referred to our paper [8] for details.)

This third class focuses on a critical threshold parameter ϵ. Items of size below ϵ are ignored and for the interval $(\epsilon, \frac{1}{2}]$, a constant value $\delta = 1/\lfloor \epsilon^{-1} \rfloor$ is used. On $(\frac{1}{2}, 1]$, the value $U^{(\epsilon)}(x_i)$ of an item x_i accounts for the number of items of size δ that can be combined with x_i Fig. 3 shows that for small values of ϵ, the area of loss zones for $\phi^{(\epsilon)}$ exceeds the area of win zones by a clear margin. This contrasts to the behavior of the functions $u^{(k)}$ and $U^{(\epsilon)}$, where the win and loss areas have the same size.

Theorem 4. *Let* $\epsilon \in [0, \frac{1}{2})$. *Then*

$$\phi^{(\epsilon)} : [0,1] \to [0,1]$$

$$x \mapsto \begin{cases} 1 - \frac{\lfloor (1-x)\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor} & \text{for } x > \frac{1}{2} \\ \frac{1}{2} & \text{for } x = \frac{1}{2} \\ \frac{1}{\lfloor \epsilon^{-1} \rfloor} & \text{for } \epsilon \leq x < \frac{1}{2} \\ 0 & \text{for } x < \epsilon \end{cases}$$

is a dual feasible function.

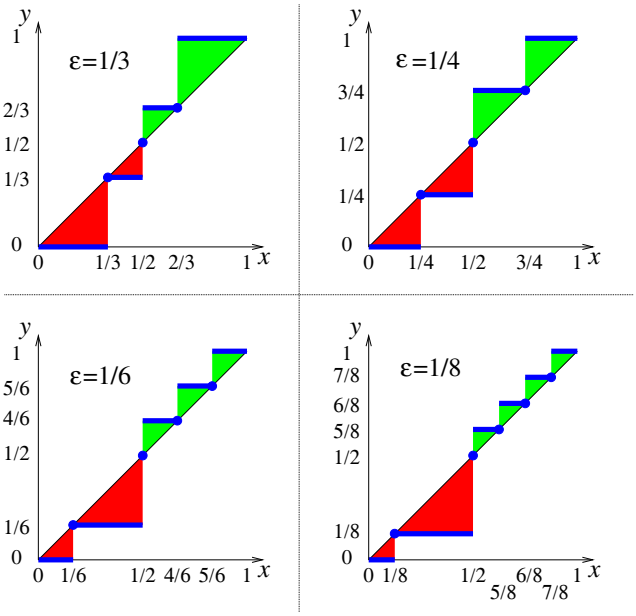

Fig. 3. Win and loss zones for $\phi^{(\epsilon)}$

Proof. Let S be a finite set of nonnegative numbers with $\sum_{x\in S} x \le 1$. Let $S' := \{x \in S \mid \epsilon \le x < \frac{1}{2}\}$. We distinguish three cases, depending on the size of the largest item:

(i) If all elements of S have size less than $\frac{1}{2}$, then by definition of S',

$$1 \ge \sum_{x\in S} x \ge \sum_{x\in S'} x \ge |S'|\epsilon \tag{14}$$

holds. Since $|S'|$ is integral, it follows that $|S'| \le \lfloor \epsilon^{-1} \rfloor$, hence

$$\sum_{x\in S} \phi^{(\epsilon)}(x) = \sum_{x\in S'} \phi^{(\epsilon)}(x) = |S'| \frac{1}{\lfloor \epsilon^{-1} \rfloor} \le 1. \tag{15}$$

(ii) The same line of argument can be applied if there are one or two items of size equal to $\frac{1}{2}$.

(iii) Finally, if S contains exactly one element $y > \frac{1}{2}$, then we have

$$1 \ge \sum_{x\in S} x \ge y + \sum_{x\in S'} x \ge y + |S'|\epsilon. \tag{16}$$

Therefore $|S'| \le \lfloor (1-y)\epsilon^{-1} \rfloor$ and hence

$$\sum_{x\in S} \phi^{(\epsilon)}(x) = \phi^{(\epsilon)}(y) + \sum_{x\in S'} \phi^{(\epsilon)}(x) = 1 - \frac{\lfloor (1-y)\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor} + |S'| \frac{1}{\lfloor \epsilon^{-1} \rfloor} \le 1. \tag{17}$$

□

4. New classes of lower bounds

In this section, we show how dual feasible functions can be combined by virtue of Lemma 2 in order to get good bounds for the BPP. We start by pointing out that the bound L_2 can easily be formulated with dual feasible functions:

Observation 2. $L_2(I) = \max_{\epsilon\in[0,\frac{1}{2}]} \left(U^{(\epsilon)}(I)\right)$.

Besides being shorter, this formulation has two other important advantages: It combines the lower bound L_1 with the dual feasible function $U^{(\epsilon)}$, so we know by virtue of Lemma 2 and Theorem 3 without any further work that this is indeed a lower bound. Moreover, this suggests an easy way of improving L_2: If we simply use additional dual feasible functions, the validity of the resulting generalized bounds is immediate.

4.1. The new class $L_*^{(p)}$

Our approach focuses at the easiest apparent weakness of L_2: As described above, having all items slightly larger than $1/3$ is a class of worst-case instances for L_2. Clearly, these instances are not really difficult, and it would be desirable to find an improvement that fares better on these easy instances. Rather than using a matching (which is still computationally expensive), we achieve an improvement in the following way:

Define for any $k \in \mathbb{N}$

$$L_2^{(k)}(I) := \max_{\epsilon\in[0,\frac{1}{2}]} L_1(u^{(k)} \circ U^{(\epsilon)}(I)) \tag{18}$$

and for $q \geq 2$ consider the following bounds:

$$L_*^{(p)}(I) := \max\left\{L_2(I), \max_{k=2,\ldots,p} L_2^{(k)}(I)\right\}. \tag{19}$$

By Lemma 2, these are valid lower bounds. Again the time needed for computing these bounds is dominated by sorting: For any particular value k, we can compute $L_2^{(k)}(I)$ in a single sweep over increasing ϵ, so we can compute this bound in time $O(n)$ for sorted items, just like the bound L_2.

Lemma 3. *Let $I := (x_1, \ldots, x_n)$ be an instance of the BPP. If the items x_i are given sorted by size, then $L_*^{(p)}(I)$ can be computed in time $O(n)$ for any fixed $p \geq 2$.*

4.2. An optimality result for $L_*^{(p)}$

Next we examine the performance of these new bounds on the worst-case instances for L_2. As a matter of fact, we have:

Theorem 5. *Let $I := (x_1, \ldots, x_n)$ be a BPP instance with all items larger than $\frac{1}{3}$. Then $L_*^{(2)}(I)$ equals the optimal value $OPT(I)$.*

Proof. Without loss of generality, let $x_1 \geq x_2 \geq \cdots \geq x_n$. Consider an optimal solution, i. e., an assignment of all items to $m := \mathrm{OPT}(I)$ bins. Obviously, any bin contains one or two items. In each bin, we call the larger item the *lower* item, while (in the bins with two items) the smaller item is called the *upper* item. In the following three steps, we transform the optimal solution into normal form.

1. Sort the bins in decreasing order of the lower item.
2. Move the upper items into the bins with highest indices. The capacity constraint remains valid, since increasing bin index corresponds to decreasing size of the lower item.
3. Using appropriate swaps, make sure that in bins with two items, increasing index corresponds to increasing size of the upper item. Again, the order of the bins guarantees that no capacity constraint is violated.

Eventually, we get the normal form shown in Fig. 4: Item x_i is placed in bin i. The remaining items $m+1, m+2, \ldots, n$ are placed in bins $m, m-1, \ldots, m-(n-(m+1))$. The first bin with two items has index $q := 2m + 1 - n$.

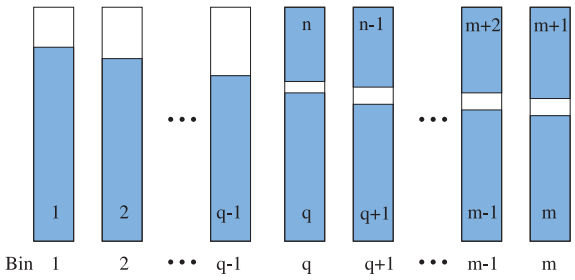

Fig. 4. Normal form of an optimal solution for a BPP instance with sizes $x_i > 1/3$

For $i \in \{1, \ldots, n\}$, the assumption $x_i > \frac{1}{3}$ implies

$$u^{(2)}(x_i) = \frac{\lfloor 3x_i \rfloor}{2} \geq \frac{1}{2}. \tag{20}$$

For $n \geq 2m - 1$, we get

$$L_2^{(2)}(I) \geq L_1(u^{(2)}(I)) = \left\lceil \sum_{i=1}^{n} u^{(2)}(x_i) \right\rceil \geq \left\lceil \frac{(2m-1)}{2} \right\rceil = m. \tag{21}$$

Hence, the lower bound $L_2^{(2)}(I)$ yields the optimal value, leaving the case

$$n < 2m - 1. \tag{22}$$

Then $q = 2m + 1 - n > 2m + 1 - (2m - 1) = 2$, implying that at least bins 1 and 2 contain only one item.

If $x_m > \frac{1}{2}$, then $x_i > \frac{1}{2}$ holds for all $i \leq m$, as the items are sorted by decreasing size, thus $U^{(1/2)}(x_i) = 1$. Hence,

$$L_2(I) \geq L_1(U^{(1/2)}(I)) \geq \left\lceil \sum_{i=1}^{m} U^{(1/2)}(x_i) \right\rceil = m. \tag{23}$$

In this case, $L_2(I)$ equals the optimal value. Therefore, assume

$$x_m \leq \frac{1}{2} \tag{24}$$

for the rest of the proof.

If $x_{m-1} + x_m > 1$, let $\epsilon := x_m \leq 1/2$. For $i \leq m-1$, we have $x_i \geq x_{m-1} > 1 - x_m = 1 - \epsilon$, hence

$$u^{(2)} \circ U^{(\epsilon)}(x_i) = u^{(2)}(1) = 1. \tag{25}$$

Furthermore, $1/3 < x_m \leq 1/2$ implies

$$u^{(2)} \circ U^{(\epsilon)}(x_m) = u^{(2)}(x_m) = \frac{1}{2}. \tag{26}$$

All in all, we have

$$L_2^{(2)}(I) \geq \left\lceil \sum_{i=1}^{m} u^{(2)} \circ U^{(\epsilon)}(x_i) \right\rceil = \left\lceil (m-1) + \frac{1}{2} \right\rceil = m, \tag{27}$$

i. e., the bound meets the optimum.

This leaves the case

$$x_{m-1} + x_m \leq 1. \tag{28}$$

Using the assumptions (22) and (28), we show that there is an $i^* \in \{2m-n, \ldots, n\}$ with

$$x_{i^*} + x_{2m-i^*-1} > 1. \tag{29}$$

Figure 5 shows the meaning of this statement: At least one of the upper items cannot be combined with the lower item two bins to the left.

This is shown in the following way: If for all $i^* \in \{2m-n, \ldots, n\}$, we had

$$x_i + x_{2m-i-1} \leq 1, \tag{30}$$

then all upper items could be moved two bins to the left, since the first two items do not contain more than one item by (22). *This would allow it to pack item x_{m-1} with x_m, saving a bin.*

Therefore, consider $\epsilon := x_{2m-i^*-1}$. By (29), we have for all $i \in \{1, \ldots, i^*\}$ that

$$x_i \geq x_{i^*} > 1 - \epsilon, \tag{31}$$

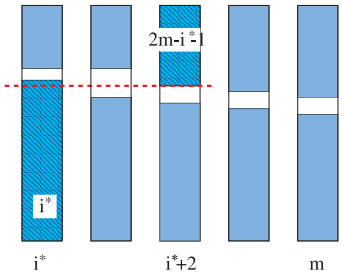

Fig. 5. Determining i^*

hence

$$u^{(2)} \circ U^{(\epsilon)}(x_i) = u^{(2)}(1) = 1. \tag{32}$$

For $i \in \{i^* + 1, \ldots, 2m - 1 - i^*\}$ we have

$$x_i \geq x_{2m-i^*-1} = \epsilon, \tag{33}$$

and therefore

$$u^{(2)} \circ U^{(\epsilon)}(x_i) \geq u^{(2)}(x_i) \geq \frac{1}{2}. \tag{34}$$

Summarizing, we get

$$L_2^{(2)}(I) \geq \left\lceil \sum_{i=1}^{i^*} u^{(2)} \circ U^{(\epsilon)}(x_i) + \sum_{i=i^*+1}^{2m-i^*-1} u^{(2)} \circ U^{(\epsilon)}(x_i) \right\rceil \tag{35}$$

$$\geq \left\lceil \sum_{i=1}^{i^*} 1 + \sum_{i=i^*+1}^{2m-i^*-1} \frac{1}{2} \right\rceil \tag{36}$$

$$= \left\lceil i^* + \frac{(2m - 2i^* - 1)}{2} \right\rceil = m. \tag{37}$$

This completes the proof.

□

4.3. Worst-case performance of $L_*^{(p)}$

As we have stated above, there is little hope to improve the absolute worst-case performance than the 2/3 achieved by L_2, since this would require solving the problem PARTITION. However, we can show that the *asymptotic* worst-case performance of L_2 is improved by $r_\infty(L_*^{(2)})$:

Theorem 6.

$$r_\infty(L_*^{(2)}) = \frac{3}{4}. \tag{38}$$

Proof. Let $I := (x_1, \ldots, x_n)$ be a BPP instance. We start by showing

$$\max\{L_2(I), L_2^{(2)}(I)\} \geq \frac{3}{4}\mathrm{OPT}(I) - 1. \tag{39}$$

By Theorem 5, all items with $x_i > \frac{1}{3}$ fit into $m \leq L_2^{(2)}(I)$ bins. Let these bins be indexed by $1, \ldots, m$. (See Fig. 6.)

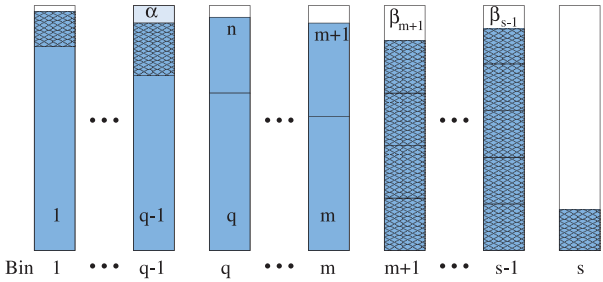

Fig. 6. Analyzing $L_*^{(2)}$

Using the *First Fit Decreasing* heuristic, we add the remaining items, i. e., sort these items by decreasing order and put each item into the first bin with sufficient capacity; if necessary, use a new bin. Let s denote the number of bins in this solution.

Suppose we need m bins for the big items and not more than $\frac{m}{3}$ new bins for the rest, then Theorem 5 yields the first part of the statement:

$$\mathrm{OPT}(I) \leq s \leq \frac{4}{3}m \leq \frac{4}{3}\max\left\{L_2^{(2)}(I), L_2(I)\right\}. \tag{40}$$

Therefore, assume

$$\frac{3s}{4} > m \tag{41}$$

for the rest of the proof.

Let α denote the largest free capacity of one of the bins 1 through m, i. e., the total size of all items in these bins is at least $(1 - \alpha)$.

No item that was placed in one of the bins $m + 1$ through s can fit into the bin with free capacity α. This means that all these bins can only contain items $x_i > \alpha$. On the other hand, the size of these items does not exceed $\frac{1}{3}$. This implies that the bins $m + 1$ through $s - 1$ must contain at least three items of size $x_i > \alpha$, while bin s holds at least one item $x_i > \alpha$.

Thus, we get

$$\begin{aligned} L_2(I) \geq L_1(I) = \left\lceil \sum_{i=1}^{n} x_i \right\rceil &\geq \lceil (1-\alpha)m + 3\alpha(q-1-m) + \alpha \rceil \\ &= \lceil (1-4\alpha)m + 3\alpha(q-1) + \alpha \rceil . \end{aligned} \tag{42}$$

Now consider two cases.

Case 1. Assume $\alpha > \frac{1}{4}$. Since $(1 - 4\alpha) < 0$, we can replace the term m in (42) by $\frac{3q}{4} > m$:

$$L_2(I) > \left\lceil (1-4\alpha)\frac{3}{4}q + 3\alpha(q-1) + \alpha \right\rceil \tag{43}$$

$$= \left\lceil \frac{3}{4}q - 2\alpha \right\rceil \tag{44}$$

$$\geq \frac{3}{4}q - 1 \geq \frac{3}{4}\mathrm{OPT}(I) - 1. \tag{45}$$

Case 2. For $\alpha \leq \frac{1}{4}$, consider the free capacities β_j of the bins $j \in \{m+1, \ldots, q-1\}$. Since no item in bin q can have size $x_i > \frac{1}{3}$, we conclude that $\beta_j \leq \frac{1}{3}$. If $\beta_j \leq \frac{1}{4}$ for all $j \in \{m+1, \ldots, q-1\}$, then $L_2(I) \geq L_1(I) \geq \frac{3}{4}(q-1) \geq \frac{3}{4}\mathrm{OPT}(I) - 1$.

Otherwise, let β_{j^*} be the first of these free capacities that is larger than $\frac{1}{4}$. Then all items in bins $\{j^*, \ldots, q-1\}$ must have size $x_i > \beta_{j^*} > \frac{1}{4}$. Since each of these bins contains at least three items, we get $\beta_j < \frac{1}{4}$ for $j \in \{j^*+1, \ldots, q-1\}$. Bin q contains at least one item of size $x_i > \beta_{j^*} > \frac{1}{4}$. Then $L_2(I) \geq L_1(I) > \frac{3}{4}(q-2) + \frac{2}{3} + \frac{1}{4} > \frac{3}{4}\mathrm{OPT}(I) - 1$.

This completes the proof that $r_\infty(L_*^{(2)}) \geq \frac{3}{4}$. For showing that equality holds, consider the family of bin packing instances with $3k$ items of size $1/4 + \delta$ with $\delta > 0$. This needs at least k bins. For sufficiently small δ, we have $L_2^{(2)} = 0$ and $L_2(I) = L_1(I) \leq \frac{3}{4}k + 1$.

□

Using a larger subset of this family of dual feasible functions does not improve the worst-case performance:

Theorem 7.

$$r_\infty(L_*^{(p)}) = \frac{3}{4}. \tag{46}$$

Proof. By Theorem 6, it is clear that $r_\infty(L_*^{(p)}) \geq \frac{3}{4}$. To see that 3/4 is still best possible, consider the following class of instances I:

For arbitrary k and sufficiently small ε, suppose we have $3k$ "small" items of size $\frac{1}{3} - \varepsilon$, and $6k$ "large" items of size $\frac{1}{3} + \varepsilon$. Then the optimal number of bins is $4k$: Each bin containing a large item can fit at most one more item; moreover, at most three small items can fit into the same bin.

On the other hand, we get $L_1(I) = L_2(I) = \lceil 3k + 15\varepsilon \rceil$. Therefore, consider rounding items sizes to multiples of $1/j$ by using the function $u^{(j)}$. As long as $j+1$ is not a multiple of 3, all sizes are rounded like items of size $1/3$, so we get overall length $3k$.

This leaves the case $j = 3i + 2$. Then small items are slightly shorter than the threshold $\frac{i+1}{3i+3} = \frac{1}{3}$, so they are rounded to $\frac{i}{3i+2}$. On the other hand, large items are rounded to $\frac{i+1}{3i+2}$. Overall, this yields a total length of $3k$.

□

The argument for the optimal value in the above example suggests the class $U^{(\epsilon_0,\dots,\epsilon_i)}$ of dual feasible functions that generalizes the class $U^{(\epsilon)}$ described in Theorem 3. That class can be interpreted in the following way: Determine an $\epsilon = \epsilon_0 > 0$, such that no item larger than $1 - \epsilon_0$ can be combined with any other item. Similarly, we can choose $\epsilon_i > 0$, such that no item beyond $1 - \epsilon_i$ can be combined with more than i other items.

Computing this bound gets more expensive, so we do not pursue it any further at this point. (See the discussion at the end of Sect. 5.2 for practical implications.)

5. Computational performance of lower bounds

Analyzing the worst-case performance of a lower bound is only one aspect of its usefulness in practice. This is best illustrated by considering the most popular way for achieving feasible solutions to the BPP: In the heuristic FIRST FIT DECREASING (FFD), items are sorted by decreasing size, and then packed greedily into the bins. It was shown by Johnson et al. [13,14] that the value $FFD(I)$ of this heuristic never exceeds $\frac{11}{9}OPT+4$. This implies that $\frac{9}{11}(FFD(I) - 4)$ is a valid lower bound. However, this lower bound is hardly useful when there is hope that a particular solution obtained by FFD is close to being optimal.

This has been observed before on predecessors of our bounds. Generally speaking, L_2 yields results that are much better than the worst-case performance (see [18,19] for a comparison with our results).

In the following, we compare the computational performance of $L_*^{(p)}$ on randomly generated benchmark problems[1]. These experiments were carried out in manner similar to the ones by Martello and Toth for the bound L_2, but we used a greater variance of parameters and a larger number of instances. As it turns out, we get a clear improvement over L_2.

5.1. Description of experiments

For our computational investigation, we consider random instances in a similar way as described in [19], pp. 240. For a given number n of items, the sizes were generated randomly with even distribution on the sets $S_1 = \{1,\dots,100\}$, $S_2 = \{1,\dots,90\}$, $S_3 = \{1,\dots,80\}$, $S_4 = \{20,\dots,80\}$, $S_5 = \{20,\dots,70\}$, for container size $c = 100$. This choice differs in part from the choice of Martello and Toth, who use the sets S_1, $S_{2'} = \{20,\dots,100\}$ and $S_{3'} = \{50,\dots,100\}$, in combination with container sizes $c = 100, 120, 150$. Our choice was motivated by the following reasons: First, it is clear that the set $S_{3'}$ is not really interesting, as it is basically covered by Theorem 5. Second, for $S_{2'}$, all items of size at least $c - 19$ cannot be combined with any other items. Choosing $\epsilon > 19/c$ for $U^{(\epsilon)}$ guarantees that all items greater or equal to $c - 19$ are indeed given weight 1. Thus, the interesting part is to consider

[1] It should be noted that the instances in the ORLIB are *not* useful as a good benchmark, since they are not challenging enough: For the 160 instances in that library, even the simple volume bound L_1 yields the optimal value for 159 instances, and for the remaining instance, L_1 yields 102 bins instead of the optimal 103 bins. See Gent [12] for a discussion.

the set $\{20, \ldots, c - 20\}$; in terms of symmetry, these instances are somewhat similar to the set $\{1, \ldots, 100\}$, but trickier due to the absence of very small items to fill remaining gaps. For comparison, we show the results for the set $\{20, \ldots, 80\}$. In addition, we give results for the set $\{20, \ldots, 70\}$, which turns out to be more difficult.

For each problem class, and $n \in \{32, 100, 316, 1000\}$, we generated 1000 instances. In Tables 1, 3, 5, 6, we compare L_1 and L_2 with the bounds $L_*^{(2)}$, $L_*^{(3)}$, $L_*^{(4)}$, $L_*^{(5)}$, $L_*^{(10)}$, $L_*^{(20)}$, $L_*^{(100)}$. Shown is the average relative error in percent (Table 1), the number of instances (out of 1000) where lower bound and upper bound coincide (Table 3), the maximum absolute gap between lower bound and upper bound (Table 5), and the total absolute gap for 1000 instances (Table 6). In addition, Tables 2 and 4 give a comparison for the bounds L_1 and L_2 with $L_1(u^{(2)})$, $L_2^{(2)}$, and $L_*^{(2)} = \max(L_2, L_2^{(2)})$.

The upper bound was computed with the routine MTP from [19] with a limit of 100000 search nodes. Note that for the trickier instances, MTP cannot be expected to find the exact optimum, and the remaining gap may be largely due to upper bound error. Furthermore, computing time for those instances was almost entirely due to running MTP for the upper bound, while the time for the lower bounds remains relatively small.

Table 1. Relative gap for lower bounds for the BPP

	Bound	L_1	L_2	$L_*^{(2)}$	$L_*^{(3)}$	$L_*^{(4)}$	$L_*^{(5)}$	$L_*^{(10)}$	$L_*^{(20)}$	$L_*^{(100)}$
Set	n									
$\{1, \ldots, 100\}$	32	5.700	0.537	0.482	0.425	0.358	0.347	0.283	0.263	0.245
	100	3.816	0.430	0.322	0.282	0.263	0.247	0.217	0.204	0.200
	316	2.325	0.280	0.186	0.160	0.148	0.142	0.118	0.109	0.108
	1000	1.391	0.184	0.114	0.101	0.094	0.091	0.078	0.074	0.072
$\{1, \ldots, 90\}$	32	5.471	0.487	0.404	0.355	0.328	0.309	0.242	0.228	0.207
	100	3.449	0.489	0.334	0.279	0.254	0.254	0.222	0.212	0.205
	316	1.663	0.240	0.145	0.121	0.111	0.109	0.100	0.094	0.093
	1000	0.667	0.097	0.041	0.034	0.032	0.031	0.028	0.027	0.027
$\{1, \ldots, 80\}$	32	3.572	0.433	0.339	0.289	0.262	0.262	0.235	0.235	0.221
	100	1.399	0.182	0.117	0.106	0.101	0.101	0.101	0.101	0.101
	316	0.240	0.041	0.022	0.022	0.022	0.022	0.022	0.022	0.022
	1000	0.009	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.002
$\{20, \ldots, 80\}$	32	7.249	0.959	0.792	0.698	0.573	0.5480	0.461	0.4195	0.401
	100	4.713	0.603	0.405	0.322	0.259	0.255	0.220	0.210	0.208
	316	2.983	0.418	0.238	0.200	0.168	0.164	0.137	0.127	0.122
	1000	1.767	0.255	0.147	0.121	0.104	0.103	0.089	0.084	0.083
$\{20, \ldots, 70\}$	32	3.387	1.526	1.446	1.341	0.864	0.800	0.735	0.696	0.636
	100	1.509	1.063	1.049	1.036	0.590	0.575	0.564	0.538	0.525
	316	1.609	1.593	1.593	1.592	1.255	1.252	1.252	1.252	1.246
	1000	1.361	1.361	1.361	1.361	1.174	1.174	1.174	1.174	1.171

5.2. Evaluation of experiments

What types of instances are easiest, which are most challenging? Quite clearly, having small items without any large items makes for the easiest type of instances, and even the simple bound L_1 does well. But even here, it is evident that our bounds squeeze out

Table 2. Comparison of L_1, L_2, $L_1(u^{(2)})$, $L_2^{(2)} = \max_{\epsilon\in[0/\frac{1}{2}]} L_1(u^{(2)} \circ U^{(\epsilon)})$, $L_*^{(2)} = \max(L_2, L_2^{(2)})$: Relative gap

	Bound	L_1	L_2	$L_1(u^{(2)})$	$L_2^{(2)}$	$L_*^{(2)}$
Set	n					
{1, . . . , 100}	32	5.700	0.537	7.885	2.199	0.482
	100	3.816	0.430	4.541	1.118	0.322
	316	2.325	0.280	2.558	0.556	0.186
	1000	1.391	0.184	1.480	0.302	0.114
{1, . . . , 90}	32	5.471	0.487	9.094	2.522	0.404
	100	3.449	0.489	5.404	1.310	0.334
	316	1.663	0.240	3.091	0.799	0.145
	1000	0.667	0.097	1.918	0.543	0.041
{1, . . . , 80}	32	3.572	0.433	11.425	3.965	0.339
	100	1.399	0.182	7.909	3.339	0.117
	316	0.240	0.041	6.295	3.537	0.022
	1000	0.009	0.003	5.967	4.357	0.003
{20, . . . , 80}	32	7.249	0.959	8.820	2.260	0.792
	100	4.713	0.603	5.196	1.159	0.405
	316	2.983	0.418	3.063	0.567	0.238
	1000	1.767	0.255	1.805	0.324	0.147
{20, . . . , 70}	32	3.3874	1.526	14.702	6.496	1.446
	100	1.509	1.063	12.329	7.116	1.049
	316	1.609	1.593	12.216	9.048	1.593
	1000	1.361	1.361	11.962	10.208	1.361

Table 3. Zero gap (out of 1000 instances) for lower bounds for the BPP

	Bound	L_1	L_2	$L_*^{(2)}$	$L_*^{(3)}$	$L_*^{(4)}$	$L_*^{(5)}$	$L_*^{(10)}$	$L_*^{(20)}$	$L_*^{(100)}$
Set	n									
{1, . . . , 100}	32	286	908	918	928	940	942	953	956	959
	100	122	775	833	854	864	872	888	895	897
	316	29	574	706	747	764	774	811	826	828
	1000	2	382	579	613	632	638	667	678	681
{1, . . . , 90}	32	347	924	937	945	949	952	963	965	968
	100	231	770	843	869	881	881	896	901	904
	316	275	679	796	826	939	841	855	864	865
	1000	410	703	854	872	878	878	888	888	888
{1, . . . , 80}	32	596	938	952	959	963	963	967	967	969
	100	672	922	950	955	957	957	957	957	957
	316	862	951	971	971	971	971	971	971	971
	1000	982	989	989	989	989	989	989	989	989
{20, . . . , 80}	32	187	836	866	882	902	906	921	928	931
	100	76	693	794	837	868	870	888	893	894
	316	21	474	676	720	755	760	795	808	814
	1000	1	354	607	646	666	666	687	698	699
{20, . . . , 70}	32	541	764	777	793	867	877	887	893	902
	100	440	528	535	541	731	738	743	755	761
	316	9	13	13	13	104	105	105	105	105
	1000	0	0	0	0	6	6	6	6	6

a considerable part of the remaining gap. For the opposite case of only relatively "bulky" items we get an excellent improvement by our new bounds, which can be seen from our experiments as well as from Theorem 5. (This does not leave much room for even better fast lower bounds, and the dual feasible function $\Phi^{(\epsilon)}$ described in Theorem 4 fails to provide such an improvement for instances with uniform distribution.)

Table 4. Comparison of L_1, L_2, $L_1(u^{(2)})$, $L_2^{(2)} = \max_{\epsilon \in [0/\frac{1}{2}]} L_1(u^{(2)} \circ U^{(\epsilon)})$, $L_*^{(2)} = \max(L_2, L_2^{(2)})$: Zero gap instances (out of 1000)

	Bound	L_1	L_2	$L_1(u^{(2)})$	$L_2^{(2)}$	$L_*^{(2)}$
Set	n					
$\{1, \ldots, 100\}$	32	286	908	151	700	918
	100	122	775	80	630	833
	316	29	574	67	604	706
	1000	2	382	42	565	579
$\{1, \ldots, 90\}$	32	347	924	159	691	937
	100	231	770	76	635	843
	316	275	679	38	552	796
	1000	410	703	39	477	854
$\{1, \ldots, 80\}$	32	596	938	129	615	952
	100	672	922	50	422	950
	316	862	951	5	152	971
	1000	982	989	0	9	989
$\{20, \ldots, 80\}$	32	187	836	160	696	866
	100	76	693	87	649	794
	316	21	474	69	629	676
	1000	1	354	49	603	607
$\{20, \ldots, 70\}$	32	541	764	45	372	777
	100	440	528	1	108	535
	316	9	13	0	4	13
	1000	0	0	0	0	0

Table 5. Maximum absolute gap between lower bounds and upper bound

	Bound	L_1	L_2	$L_*^{(2)}$	$L_*^{(3)}$	$L_*^{(4)}$	$L_*^{(5)}$	$L_*^{(10)}$	$L_*^{(20)}$	$L_*^{(100)}$
Set	n									
$\{1, \ldots, 100\}$	32	5	1	1	1	1	1	1	1	1
	100	11	1	1	1	1	1	1	1	1
	316	14	3	2	2	2	2	1	1	1
	1000	27	5	4	4	3	3	2	2	2
$\{1, \ldots, 90\}$	32	5	1	1	1	1	1	1	1	1
	100	10	2	1	1	1	1	1	1	1
	316	17	4	2	2	1	1	1	1	1
	1000	24	5	3	3	3	3	3	2	2
$\{1, \ldots, 80\}$	32	5	1	1	1	1	1	1	1	1
	100	7	2	1	1	1	1	1	1	1
	316	9	2	1	1	1	1	1	1	1
	1000	9	1	1	1	1	1	1	1	1
$\{20, \ldots, 80\}$	32	5	1	1	1	1	1	1	1	1
	100	13	2	1	1	1	1	1	1	1
	316	19	4	3	3	2	2	2	2	2
	1000	34	8	5	5	4	4	4	4	4
$\{20, \ldots, 70\}$	32	8	1	1	1	1	1	1	1	1
	100	7	2	2	2	2	2	2	2	2
	316	11	6	6	6	5	5	5	5	5
	1000	12	12	12	12	12	12	12	12	12

There is one scenario where or bounds provide only little gain over L_1 and L_2: This is the situation where we have neither small nor really large items, so there is no way to avoid larger empty space in bins, but also no way to account for it by focusing on individual large items. This is not surprising, as the forced empty space is due to combinations of items. (Here too, $\Phi^{(\epsilon)}$ cannot yield an improvement, since its gain is limited to larger items.) There is one consolation in having to deal with a larger

Table 6. Total absolute gap in bins for 1000 instances

	Bound	L_1	L_2	$L_*^{(2)}$	$L_*^{(3)}$	$L_*^{(4)}$	$L_*^{(5)}$	$L_*^{(10)}$	$L_*^{(20)}$	$L_*^{(100)}$
Set	n									
{1, . . . , 100}	32	1080	92	82	72	60	58	47	44	41
	100	2109	225	167	146	136	128	112	105	103
	316	3899	454	299	257	237	227	189	174	172
	1000	7222	938	580	513	475	461	394	374	363
{1, . . . , 90}	32	956	76	63	55	51	48	37	35	32
	100	1737	232	157	131	119	119	104	99	96
	316	2533	353	211	176	161	159	145	136	135
	1000	3129	448	190	158	145	144	129	126	126
{1, . . . , 80}	32	572	62	48	41	37	37	33	33	31
	100	639	79	50	45	43	43	43	43	43
	316	329	54	29	29	29	29	29	29	29
	1000	36	11	11	11	11	11	11	11	11
{20, . . . , 80}	32	1376	164	134	118	98	94	79	72	69
	100	2598	311	206	163	132	130	112	107	106
	316	4993	673	378	318	267	260	218	201	194
	1000	9128	1289	738	605	520	513	443	421	413
{20, . . . , 70}	32	565	236	223	207	133	123	113	107	98
	100	722	495	488	482	273	266	261	249	243
	316	2338	2313	2313	2311	1818	1814	1814	1813	1805
	1000	6213	6213	6213	6213	5355	5355	5355	5355	5345

minimum number of items per bin: Even the bound L_1 must do considerably better than its worst-case performance.

To give a clearer idea of the limitations of the various linear-time bounds in this scenario, we show the results for instances with 100 items and the sets {20, . . . , 80} {20, . . . , 70}, {20, . . . , 60}, {20, . . . , 50}, {20, . . . , 40}, {20, . . . , 35}, {20, . . . , 30}, in Table 7.

The most difficult instances seem to occur when using the interval {20, . . . , 35}. This coincides with the observation by [22]. Clearly, there is little hope for accounting for the empty space forced by the absence of "good" combinations of several items without going through those combination. In principle, it is possible to get better results for these instances by evaluating subsets of items, instead of individual ones; this can be considered as the column generation bound for a subset of feasible sets. It should be possible to get lower bounds for these cases by using appropriate types of dual feasible functions, as mentioned at the end of Sect. 4. Since the main objective of this paper is to get get lower bound in near-linear time, this extension is left to future work.

6. Conclusions

We have presented a new method for generating fast lower bounds for the bin packing problem, and demonstrated that this approach provides useful results. The underlying method of dual feasible functions can also be used in the case of several dimensions by combining our ideas with the approach for modeling multi-dimensional orthogonal packings that we developed for finding exact solutions for the d-dimensional knapsack problem [6]. Details are described in the papers [7–9].

Table 7. Performance for instances without large or small items

$n = 100$	Bound	L_1	L_2	$L_*^{(2)}$	$L_*^{(3)}$	$L_*^{(4)}$	$L_*^{(5)}$	$L_*^{(10)}$	$L_*^{(20)}$	$L_*^{(100)}$
Set										
{20, ..., 80}	rel. error	4.713	0.603	0.405	0.322	0.259	0.255	0.220	0.210	0.208
	zero gap	76	693	794	837	868	870	888	893	894
	max gap	13	2	1	1	1	1	1	1	1
	total gap	2598	311	206	163	132	130	112	107	106
{20, ..., 70}	rel. error	1.509	1.063	1.049	1.036	0.590	0.575	0.564	0.538	0.525
	zero gap	440	528	535	541	731	738	743	755	761
	max gap	7	2	2	2	2	2	2	2	2
	total gap	722	495	488	482	273	266	261	249	243
{20, ..., 60}	rel. error	1.340	1.340	1.340	1.324	1.324	1.324	1.324	1.324	1.324
	zero gap	483	483	483	484	484	484	484	484	484
	max gap	3	3	3	3	3	3	3	3	3
	total gap	556	556	556	549	549	549	549	549	549
{20, ..., 50}	rel. error	4.321	4.321	4.321	4.321	4.318	4.318	4.318	4.318	4.214
	zero gap	56	56	56	56	57	57	57	57	60
	max gap	3	3	3	3	3	3	3	3	3
	total gap	1615	1615	1615	1615	1614	1614	1614	1614	1575
{20, ..., 40}	rel. error	2.236	2.236	2.236	2.236	2.236	2.236	2.236	2.236	2.236
	zero gap	363	363	363	363	363	363	363	363	363
	max gap	2	2	2	2	2	2	2	2	2
	total gap	702	702	702	702	702	702	702	702	702
{20, ..., 35}	rel. error	6.020	6.020	6.020	6.020	6.020	6.020	6.020	6.020	6.016
	zero gap	4	4	4	4	4	4	4	4	4
	max gap	3	3	3	3	3	3	3	3	3
	total gap	1796	1796	1796	1796	1796	1796	1796	1796	1795
{20, ..., 30}	rel. error	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.594
	zero gap	603	603	603	603	603	603	603	603	603
	max gap	3	3	3	3	3	3	3	3	3
	total gap	727	727	727	727	727	727	727	727	726

Acknowledgements. We thank an anonymous referee for several helpful comments that helped to improve the overall presentation of this paper. In particular, Sect. 5 has benefited greatly from these suggestions.

References

1. Beasley, J.E. (1990): OR-Library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**, 1069–1072, `http://mscmga.ms.ic.ac.uk/info.html`
2. Coffmann, Jr., E.G., Garey, M.R., Johnson, D.S. (1997): Approximation algorithms for bin packing: a survey. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-hard Problems. PWS Publishing, Boston, pp. 46–93
3. Chan, L.M.A., Simchi-Levi, D., Bramel, J. (1998): Worst-case analyses, linear programming and the bin-packing problem. Math. Program. **83**, 213–227
4. Coffmann, Jr., E.G., Lueker, G.S. (1991): Probabilistic Analysis of Packing and Partitioning Algorithms. Wiley, New York
5. Fekete, S.P., Schepers, J. (1998): New classes of lower bounds for bin packing problems. Proceedings of the 6th International Integer Programming and Combinatorial Optimization Conference. Springer Lecture Notes in Computer Science **1412**, 257–270
6. Fekete, S.P., Schepers, J. (1997): A new exact algorithm for general orthogonal d-dimensional knapsack problems. Algorithms – ESA '97, Springer Lecture Notes in Computer Science **1284**, 144–156
7. Fekete, S.P., Schepers, J.: On higher-dimensional packing I: Modeling. Technical Report 97-288. To appear in Mathematics of Operations Research.
Available at `http://www.zaik.uni-koeln.de/~paper`
8. Fekete, S.P., Schepers, J.: On higher-dimensional packing II: Bounds. Technical Report 97-289. Available at `http://www.zaik.uni-koeln.de/~paper`
9. Fekete, S.P., Schepers, J.: On higher-dimensional packing III: Exact Algorithms. Technical Report 97-290. Available at `http://www.zaik.uni-koeln.de/~paper`

10. Fernandez de la Vega, W., Lueker, G.S. (1981): Bin packing can be solved within $1 + \varepsilon$ in linear time. Combinatorica **1**, 349–355
11. Garey, M.R., Johnson, D.S. (1979): Computers and Intractability: a guide to the theory of NP-completeness. Freeman, San Francisco
12. Gent, I.P. (1998): Heuristic solution of open bin packing problems. J. Heuristics **3**, 299–304
13. Johnson, D.S. (1973): Near-optimal bin packing algorithms. Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts
14. 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 J. Comput. **3**, 299–325
15. Karmarkar, N., Karp, R.M. (1982): An efficient approximation scheme for the one-dimensional bin packing problem. Proc. 23rd Annual Symp. Found. Comp. Sci. (FOCS 1982), pp. 312–320
16. Lueker, G.S. (1983): Bin packing with items uniformly distributed over intervals $[a, b]$. Proc. 24th Annual Symp. Found. Comp. Sci. (FOCS 1983), pp. 289–297
17. Martello, S., Pisinger, D., Vigo, D. (2000): The three-dimensional bin packing problem. Oper. Res. **48**, 256–267
18. Martello, S., Toth, P. (1990): Lower bounds and reduction procedures for the bin packing problem. Discrete Appl. Math. **28**, 59–70
19. Martello, S., Toth, P. (1990): Knapsack Problems. Wiley, New York
20. Martello, S, Vigo, D. (1998): Exact solution of the two-dimensional finite bin packing problem. Manage. Sci. **44**, 388–399
21. Schepers, J. (1997): Exakte Algorithmen für orthogonale Packungsprobleme. Dissertation, Mathematisches Institut, Universität zu Köln
22. Scholl, A., Klein, R., Jürgens, C. (1997): BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Comput. Oper. Res. **24**, 627–645