Lemma 9

 $OPT(I') \le OPT(I) \le OPT(I') + k$

Proof 2:

- Any bin packing for *I*′ gives a bin packing for *I* as follows.
- Pack the items of group 1 into *k* new bins;
- Pack the items of groups 2, where in the packing for I' the items for group 2 have been packed;

▶ ...

EADS II © Harald Räcke	17.3 Bin Packing	344

Can we do better?

In the following we show how to obtain a solution where the number of bins is only

$$OPT(I) + O(log^2(SIZE(I)))$$

Note that this is usually better than a guarantee of

$$(1 + \epsilon)$$
OPT $(I) + 1$

Assume that our instance does not contain pieces smaller than $\epsilon/2$. Then SIZE(I) $\geq \epsilon n/2$.

We set $k = \lfloor \epsilon \text{SIZE}(I) \rfloor$.

Then $n/k \le 2n/\lfloor \epsilon^2 n/2 \rfloor \le 4/\epsilon^2$ (here we used $\lfloor \alpha \rfloor \ge \alpha/2$ for $\alpha \ge 1$).

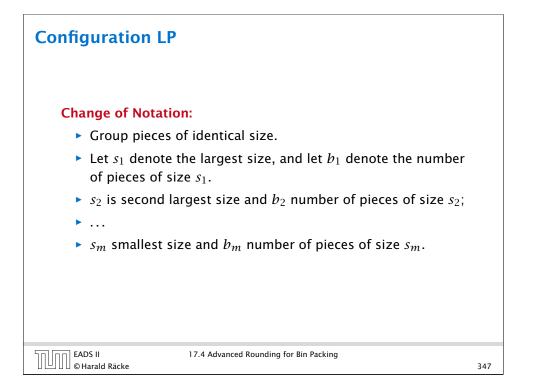
Hence, after grouping we have a constant number of piece sizes $(4/\epsilon^2)$ and at most a constant number $(2/\epsilon)$ can fit into any bin.

We can find an optimal packing for such instances by the previous Dynamic Programming approach.

cost (for large items) at most

 $OPT(I') + k \le OPT(I) + \epsilon SIZE(I) \le (1 + \epsilon)OPT(I)$

• running time $\mathcal{O}((\frac{2}{\epsilon}n)^{4/\epsilon^2})$.



Configuration LP

A possible packing of a bin can be described by an *m*-tuple (t_1, \ldots, t_m) , where t_i describes the number of pieces of size s_i . Clearly,

 $\sum_i t_i \cdot s_i \leq 1$.

We call a vector that fulfills the above constraint a configuration.

Soloo EADS II	17.4 Advanced Rounding for Bin Packing	
© Harald Räcke		348

How to solve this	EP?	
later		
EADS II © Harald Räcke	17.4 Advanced Rounding for Bin Packing	350

Configuration LP

 \mathbb{N}

Let N be the number of configurations (exponential).

Let T_1, \ldots, T_N be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).

n	nin		$\sum_{i=1}^{N} x_i$			
S	s.t.	$\forall i \in \{1 \dots m\}$	$\sum_{j=1}^{N} T_{ji} x_j$	≥	b_i	
		$\forall j \in \{1, \dots, N\}$	x_j	≥	0	
		$\forall j \in \{1, \dots, N\}$	x_{j}	integral		

We can assume that each item has size at least 1/SIZE(I).

Harmonic Grouping

- Sort items according to size (monotonically decreasing).
- Process items in this order; close the current group if size of items in the group is at least 2 (or larger). Then open new group.
- I.e., G₁ is the smallest cardinality set of largest items s.t. total size sums up to at least 2. Similarly, for G₂,...,G_{r-1}.
- Only the size of items in the last group G_r may sum up to less than 2.

	17.4 Advanced Rounding for Bin Packing	252
🛛 🕒 🛛 🖉 © Harald Räcke		352

Lemma 10

The number of different sizes in I' is at most SIZE(I)/2.

- Each group that survives (recall that G₁ and G_r are deleted) has total size at least 2.
- Hence, the number of surviving groups is at most SIZE(I)/2.
- All items in a group have the same size in I'.

EADS II © Harald Räcke

17.4 Advanced Rounding for Bin Packing

Harmonic Grouping

From the grouping we obtain instance I' as follows:

- Round all items in a group to the size of the largest group member.
- Delete all items from group G_1 and G_r .
- For groups G_2, \ldots, G_{r-1} delete $n_i n_{i-1}$ items.
- Observe that $n_i \ge n_{i-1}$.

EADS II © Harald Räcke	17.4 Advanced Rounding for Bin Packing	

Lemma 11

The total size of deleted items is at most $O(\log(SIZE(I)))$.

- ▶ The total size of items in *G*¹ and *G*^{*r*} is at most 6 as a group has total size at most 3.
- Consider a group G_i that has strictly more items than G_{i-1} .
- It discards $n_i n_{i-1}$ pieces of total size at most

$$3\frac{n_i - n_{i-1}}{n_i} \le \sum_{j=n_{i-1}+1}^{n_i} \frac{3}{j}$$

since the smallest piece has size at most $3/n_i$.

Summing over all *i* that have n_i > n_{i-1} gives a bound of at most

$$\sum_{j=1}^{l_{r-1}} \frac{3}{j} \le \mathcal{O}(\log(\text{SIZE}(I))) \quad .$$

(note that $n_r \leq \text{SIZE}(I)$ since we assume that the size of each item is at least 1/SIZE(I)).

EADS II © Harald Räcke

354

353

Algorithm 1 BinPack

- 1: **if** SIZE(I) < 10 **then**
- 2: pack remaining items greedily
- 3: Apply harmonic grouping to create instance I'; pack discarded items in at most $O(\log(SIZE(I)))$ bins.
- 4: Let x be optimal solution to configuration LP
- 5: Pack $\lfloor x_j \rfloor$ bins in configuration T_j for all j; call the packed instance I_1 .
- 6: Let I_2 be remaining pieces from I'
- 7: Pack I_2 via BinPack (I_2)

EADS II	17.4 Advanced Rounding for Bin Packing	
UUU©Harald Räcke		356

Analysis

Each level of the recursion partitions pieces into three types

- 1. Pieces discarded at this level.
- **2.** Pieces scheduled because they are in I_1 .
- **3.** Pieces in I_2 are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\mbox{OPT}_{\mbox{LP}}$ many bins.

Pieces of type 1 are packed into at most

$\mathcal{O}(\log(\text{SIZE}(I))) \cdot L$

many bins where L is the number of recursion levels.

החוחר	EADS II © Harald Räcke
	© Harald Räcke

358

Analysis

$OPT_{LP}(I_1) + OPT_{LP}(I_2) \le OPT_{LP}(I') \le OPT_{LP}(I)$

Proof:

- ► Each piece surviving in I' can be mapped to a piece in I of no lesser size. Hence, OPT_{LP}(I') ≤ OPT_{LP}(I)
- $\lfloor x_j \rfloor$ is feasible solution for I_1 (even integral).
- $x_j \lfloor x_j \rfloor$ is feasible solution for I_2 .

EADS II © Harald Räcke 17.4 Advanced Rounding for Bin Packing

Analysis

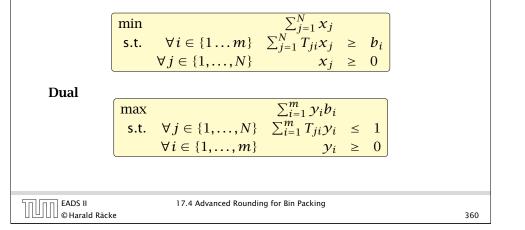
We can show that $SIZE(I_2) \le SIZE(I)/2$. Hence, the number of recursion levels is only $O(\log(SIZE(I_{original})))$ in total.

- ► The number of non-zero entries in the solution to the configuration LP for I' is at most the number of constraints, which is the number of different sizes (≤ SIZE(I)/2).
- ► The total size of items in I_2 can be at most $\sum_{j=1}^{N} x_j \lfloor x_j \rfloor$ which is at most the number of non-zero entries in the solution to the configuration LP.

How to solve the LP?

Let T_1, \ldots, T_N be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i). In total we have b_i pieces of size s_i .

Primal



Separation Oracle

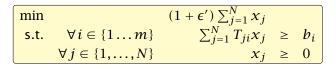
We have FPTAS for Knapsack. This means if a constraint is violated with $1 + \epsilon' = 1 + \frac{\epsilon}{1-\epsilon}$ we find it, since we can obtain at least $(1 - \epsilon)$ of the optimal profit.

The solution we get is feasible for:

Dual'

 $\begin{array}{|c|c|c|c|c|c|} \hline \max & & \sum_{i=1}^{m} y_i b_i \\ \text{s.t.} & \forall j \in \{1, \dots, N\} & \sum_{i=1}^{m} T_{ji} y_i & \leq & 1 + \epsilon' \\ & \forall i \in \{1, \dots, m\} & & y_i & \geq & 0 \end{array}$

Primal'



Separation Oracle

Suppose that I am given variable assignment y for the dual.

How do I find a violated constraint?

I have to find a configuration $T_j = (T_{j1}, \ldots, T_{jm})$ that

m

is feasible, i.e.,

$$\sum_{i=1}^m T_{ji} \cdot s_i \le 1 \quad ,$$

and has a large profit

$$\sum_{i=1}^{m} T_{ji} y_i > 1$$

But this is the Knapsack problem.

EADS II 17.4 Advanced Rounding for Bin Packing © Harald Räcke 361

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$OPT \le z \le (1 + \epsilon')OPT$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- ► Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL" is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1 + \epsilon')$ OPT.
- > We can compute the corresponding solution in polytime.

This gives that overall we need at most

$$(1 + \epsilon')$$
OPT_{LP} $(I) + O(\log^2(SIZE(I)))$

bins.

We can choose $\epsilon' = \frac{1}{OPT}$ as $OPT \le \#$ items and since we have a fully polynomial time approximation scheme (FPTAS) for knapsack.

	17.4 Advanced Rounding for Bin Packing	
UUU©Harald Räcke		364

