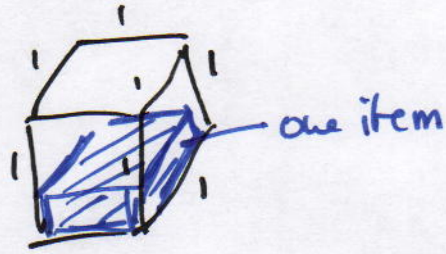
Min Bin Packing

- items with given breadth and depth 1, height ≤ 1

INTO:

- cuboids of side length 1



- $s_1, \ldots, s_n \in (0,1)$
- a <u>bin</u> B is a subset $B \subseteq \{s_1, \ldots, s_n\}$, such that

$$\sum_{s_i \in B} s_i \leq 1$$

- <u>Bin Packing</u>: partition of the set $\{s_1, \ldots, s_n\}$ into bins
- <u>MIN BIN PACKING</u>: find a partition into as few bins as possible:

INPUT: numbers $s_1, \ldots, s_n \in (0,1)$

OUTPUT: Partition $B_1, \ldots, B_k$ of $\{s_1, \ldots, s_n\}$ in bins, with k minimal

* MIN BIN PACKING is NP-hard ($\rightarrow$ lecture)

4 different approximation algorithms, input $L = (s_1, \dots, s_n)$

* <u>algorithm FF(L) (First Fit)</u>

  FOR $j=1$ TO n DO

  pack s_j in bin B_i with smallest index in which s_j fits

* <u>algorithm BF(L) (Best Fit)</u>

  FOR $j=1$ TO n DO

  pack s_j in bin B_i with smallest unused capacity in which s_j fits

* <u>algorithm FFD(L) (First Fit Decreasing)</u>

  Sort the list L, such that $s_1 \geq \dots \geq s_n$

  Apply FF

* <u>algorithm BFD(L) (Best Fit Decreasing)</u>

  Sort the list L, such that $s_1 \geq \dots \geq s_n$

  Apply BF

And how good are these algorithms?

<u>Claim 1:</u> FF and BF cannot achieve an approximation factor better than $5/3$.

↳ We give an example L, with

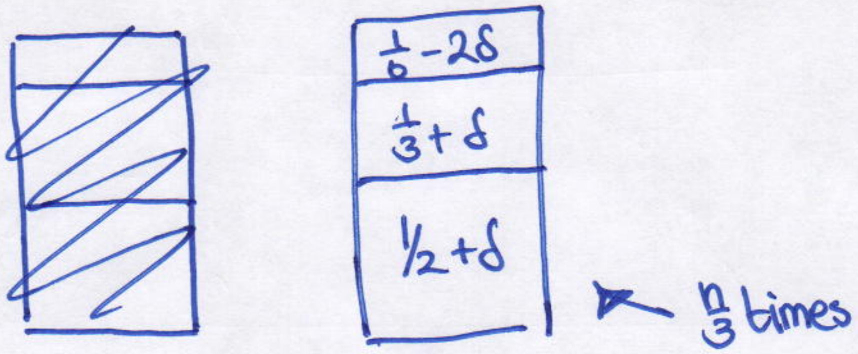
$$FF(L), BF(L) = \frac{5}{3}\, OPT(L)$$

Let n be a multiple of 18, $0 < \delta < \frac{1}{84}$.

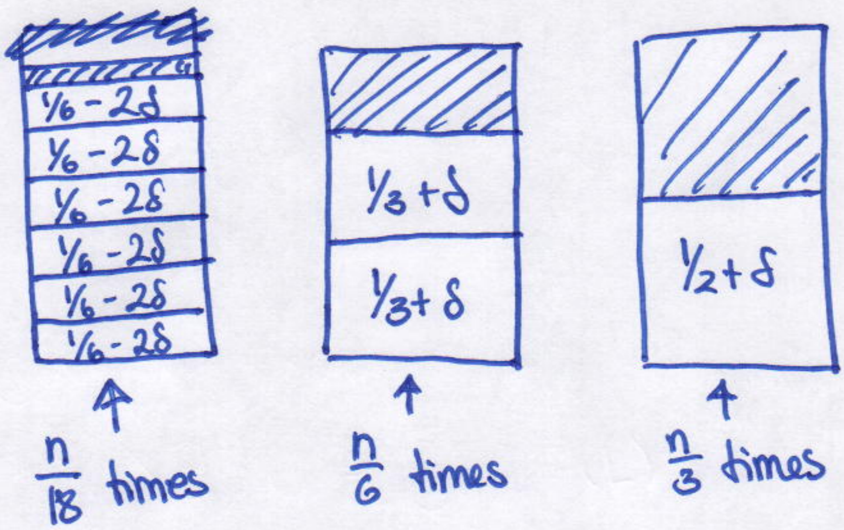
Define $L = (s_1, \dots, s_n)$ by

$$s_i = \begin{cases} \frac{1}{6} - 2\delta & , \text{ if } 1 \le i \le \frac{n}{3} \\ \frac{1}{3} + \delta & , \text{ if } \frac{n}{3} < i \le \frac{2n}{3} \\ \frac{1}{2} + \delta & , \text{ if } \frac{2n}{3} < i \le n \end{cases}$$

OPT(L)? $\quad OPT(L) = \frac{n}{3}$



Algorithms FF / BF?



$$\Rightarrow FF(L), BF(L) = \frac{n}{18} + \frac{n}{6} + \frac{n}{3} = \frac{n}{18} + \frac{3n}{18} + \frac{6n}{18}$$

$$= \frac{10n}{18} = \frac{5n}{9}$$

$$\Rightarrow FF(L), BF(L) = \frac{5n}{9} = \frac{5}{3} \cdot \underbrace{\frac{1}{3} n}_{= OPT(L)}$$

without proof:

<u>Theorem</u> [Johnson, Demers, Ullman; Garey, Graham; 1974]

For each list L:

$$FF(L), BF(L) \leq \frac{17}{10} OPT(L) + 2$$

(proof over 7 pages...)

How good can <u>any</u> approximation algorithm be?

<u>Theorem 2</u>: Provided that $P \neq NP$ no polynomial approximation algorithm for MIN BIN PACKING with an approximation factor better than $3/2$ can exist.

<u>Proof</u>: we show: if there exists a pol. time algorithm for MIN BIN PACKING with a factor better than $3/2$, the problem PARTITION can be solved in pol. time

<u>PARTITION</u>:

Input: set $S = \{a_1, ..., a_n\}$, $a_i \in \mathbb{N}$

Question: $\exists$ partition $S = S_1 \dot{\cup} S_2$, such that

$$\sum_{a_i \in S_1} a_i = \sum_{a_i \in S_2} a_i$$

T odd $\rightarrow$ no partition exists.

Assume, there is a pol. time approximation alg. $\mathcal{A}$ for MIN BIN PACKING with factor better than $3/2$. (5)

Let $S = \{a_1, \ldots, a_n\}$ be an instance of PARTITION and $T := \sum_{i=1}^{n} a_i$.

T odd $\rightarrow$ no partition exists

T even: let

$$L = \left(\frac{2 \cdot a_1}{T}, \ldots, \frac{2 \cdot a_n}{T} \right)$$

be an instance of MIN BIN PACKING an let m be the number of Bins constructed by $\mathcal{A}$ with an input of L.

We distinguish 2 cases:

* case 1: $m \geq 3$

$\hookrightarrow$ as: $\frac{m}{OPT(L)} < \frac{3}{2} \Rightarrow OPT(L) > 2$

$\Rightarrow$ items $\frac{2a_1}{T}, \ldots, \frac{2a_n}{T}$ cannot be packed in 2 bins of size 1

In particular, the input $S = \{a_1, \ldots, a_n\}$ of PARTITION is a NO-instance

* case 2: $m \leq 2$

as $\sum_{i=1}^{n} \frac{2a_i}{T} = 2 \Rightarrow m = 2$

$\Rightarrow$ both bins packed by $\mathcal{A}$ are full packed

$\Rightarrow$ Partition der Menge S

$\hookrightarrow$ YES-instance $\square$

- PTAS / FPTAS
- Knapsack

- Π: NP-hard optimization problem, obj. function f_Π
- algorithm $\mathcal{A}$ is an <u>approximation scheme</u> for Π if on input (I, ε) it outputs a solution s such that:

instance of Π error parameter

* $f_\Pi(I,s) \leq (1+\varepsilon) \cdot OPT$ if Π is a minimization probl.
* $f_\Pi(I,s) \geq (1-\varepsilon) \cdot OPT$ — " — maximization — " —

- $\mathcal{A}$ is a PTAS, a <u>polynomial time approximation scheme</u>, if for ~~every~~ each fixed $\varepsilon > 0$, its running time is bounded by a polynomial in the size of instance I.

running time of $\mathcal{A}$ can depend arbitrarily on ε

⇒ if the running time of $\mathcal{A}$ is bounded by a polynomial in the size of instance I and $1/\varepsilon$

⇒ $\mathcal{A}$ is FPTAS, a <u>fully pollynomial approximation scheme</u>

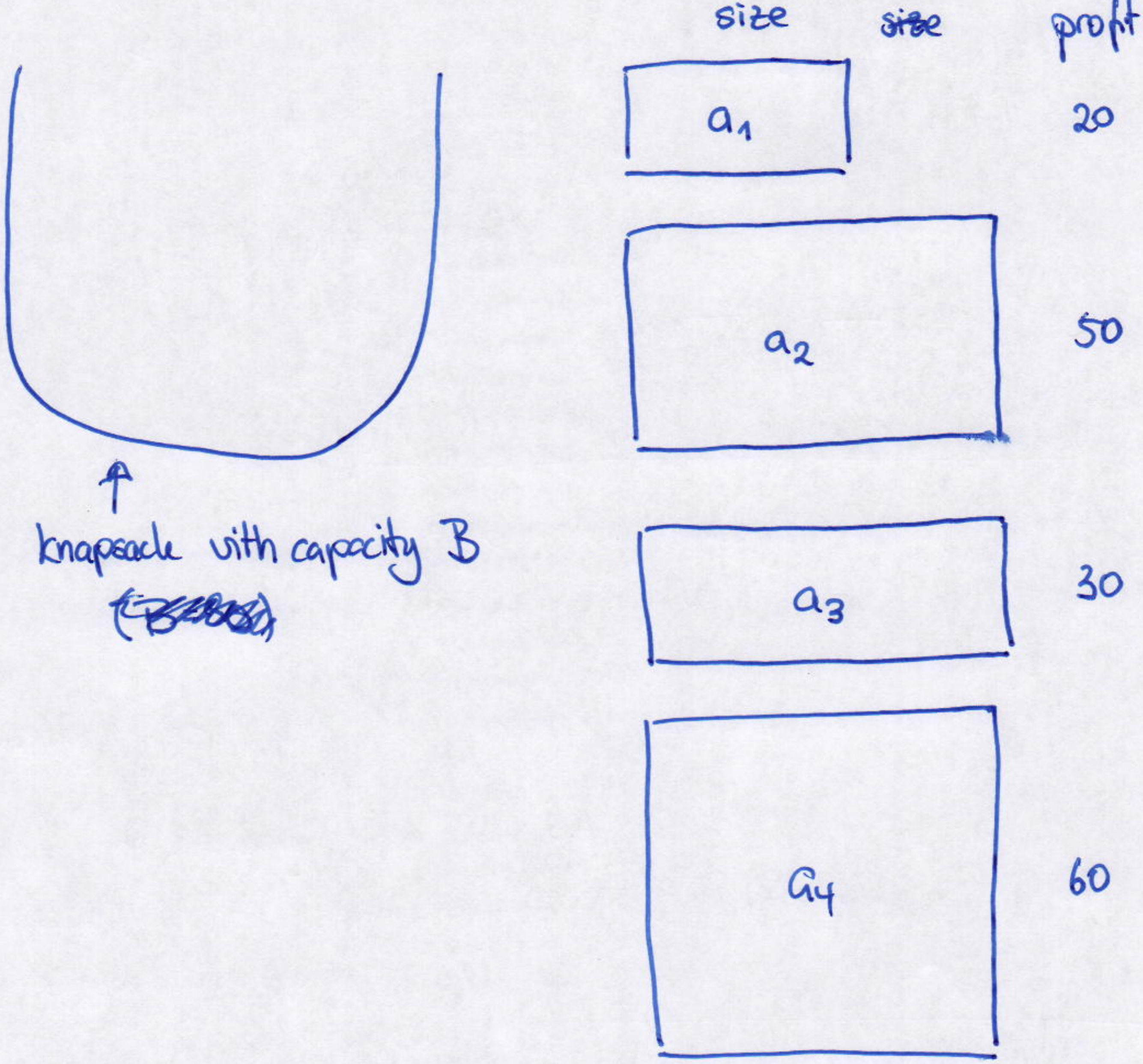
„best" for NP-hard problems

⇝ July: PTAS (guillotine subdivision) for geometric problems

Knapsack

Given: set $S = \{a_1, \dots, a_n\}$ of objects, with sizes and profits:
$size(a_i) \in \mathbb{Z}^+$, $profit(a_i) \in \mathbb{Z}^+$
„knapsack capacity" $B \in \mathbb{Z}^+$

Task: Find $K \subseteq S$ whose total size is bounded by B and total profit is maximized



Here: $K = \{a_2, a_3\}$, profit: 80

III

- first idea: greedy → sort objects in decreasing order of ratio $\frac{\text{profit}}{\text{size}}$, pick greedily
  - **
  - * does not work for (0,1)-version → homework
  - * ok if items are divisible – fractional solution

→ 1. A pseudo-polynomial time algorithm for knapsack

↳ dynamic* programming (NWA!)

So far: size of instance I, $|I|$, # of bits needed to write I, ~~when at~~ assuming all numbers to be written in binary

I_u: instance I with all numbers occuring written in unary

unary:	
0	0
1	10
2	110
⋮	

unary size of I, $|I_u|$: # bits needed to write I_u

↳ an algorithm for Π whose running time on instance I is bounded by a polynomial in $|I_u|$: pseudo-polynomial time algorithm

Let $P = \max_{a \in S} \text{profit}(a)$ ← most profitable object

↳ $n \cdot P$ upper bound

$\forall\, i \in \{1, \ldots, n\},\ p \in \{1, \ldots, nP\}$:

$S_{i,p}$: subset of $\{a_1, \ldots, a_i\}$ whose total profit is exactly p
AND whose total size is minimized

$A(i,p)$: size of $S_{i,p}$
($A(i,p) = \infty$ if no such set exists)

$A(1,p)$ known $\forall\, p \in \{1, \ldots, nP\}$

recurrence for rest:

$$A(i+1,p) = \begin{cases} \min\{A(i,p),\ \text{size}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1}))\} & ;\ \text{if } \text{profit}(a_{i+1}) < p \\ A(i,p) & ;\ \text{otherwise} \end{cases}$$

↳ computation: $O(n^2 P)$ time

max profit: $\max\{p \mid A(n,p) \leq B\}$

↳ pseudo-polynomial algorithm for knapsack

input size: $\log W$ not W → not polynomial in $|I|$

note: if profits were small numbers, i.e., bounded by a polynomial in n (*)
↳ would be a regular polynomial time algorithm