Fundamental Computer Science
Lecture 6: More on approximation
Focus on Bin Packing

Denis Trystram
MoSIG1 and M1Info – University Grenoble-Alpes

April, 2021

A full example

The idea here is to study the whole process for studying/analyzing a *complex* problem.

- Description of the problem and modelling
- Complexity study
  (In)approximation – in case
- Solving the problem:
  Heuristics and their analysis
- Performance guarantee (Polynomial time approximations)

The story

Let us imagine you have to move fast to a new place and you should store your personal effects in a limited place garage.
All your goods are packed into boxes of different sizes (same basis but with different heights).

Theoretical model

- Decision version.

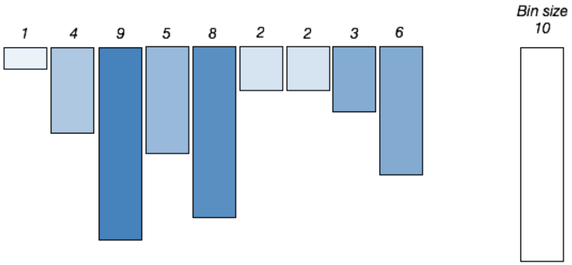
BIN-PACKING

Input: a set of items A, a size $s(a)$ for each $a \in A$, a positive integer capacity C, and a positive integer k

Question: is there a partition of A into disjoint sets $A_1, A_2, \ldots, A_k$ such that the total size of the elements in each set A_j does not exceed the capacity C, i.e., $\sum_{a \in A_j} s(a) \leq C$?

Some hypotheses: the sizes $s(a)$ are integers. No problem to extend to rational numbers.

Example: 9 items and $C = 10$



Complexity analysis

Let us first prove that BinPacking is in NP-complete.

This is easy by a simple reduction from 2Partition.
Recall the method.

1. BinPacking $\in \mathcal{NP}$
   Verifier
   - given the subset of integers packed in each of the k bins A_j, create the sum of these elements and compare with C

Complexity analysis

Let us first prove that BinPacking is in NP-complete.

This is easy by a simple reduction from 2Partition.
Recall the method.

1. BinPacking $\in \mathcal{NP}$
   Verifier
   - given the subset of integers packed in each of the k bins A_j, create the sum of these elements and compare with C
2. 2Partition $\leq_{\mathrm{P}}$ BinPacking
   This is straightforward since 2Partition is a subproblem of BinPacking (specific instances with 2 bins)

Complexity analysis

Let us first prove that BinPacking is in NP-complete.

This is easy by a simple reduction from 2Partition.
Recall the method.

1. BinPacking $\in \mathcal{NP}$
   Verifier
   - given the subset of integers packed in each of the k bins A_j, create the sum of these elements and compare with C
2. 2Partition $\leq_P$ BinPacking
   This is straightforward since 2Partition is a subproblem of BinPacking (specific instances with 2 bins)

   We can prove a deeper result (strongly NP-complete):
   3Partition $\leq_P$ BinPacking

Inapproximability

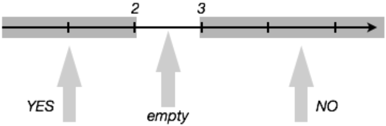
BinPacking can not be approximated by a factor better than $3/2$

- assume by contradiction that it can be approximated by a ratio $\rho < 3/2$
- apply the gap reduction to a positive instance of $< A, C, 2 >$

Inapproximability

$\textsc{BinPacking}$ can not be approximated by a factor better than $3/2$

- assume by contradiction that it can be approximated by a ratio $\rho < 3/2$
- apply the gap reduction to a positive instance of $< A, C, 2 >$
- If $OPT(\mathcal{I}) \leq 2$ then $SOL(\mathcal{I}) \leq 2 \cdot \rho < 3$ then $SOL(\mathcal{I}) = 2$



- Thus, solving this problem corresponds to solve $2\textsc{Partition}$ in polynomial time, unless $\mathcal{P} = \mathcal{NP}$

Approximation ratio: recall

- consider a minimization problem Π and a polynomial-time algorithm $\mathcal{A}$ for solving this problem
- $OPT(\mathcal{I})$: the objective value of an optimal solution for the instance $\mathcal{I}$ of the problem Π
- $SOL(\mathcal{I})$: the objective value of the solution of our algorithm $\mathcal{A}$ for the instance $\mathcal{I}$

Approximation ratio: recall

- consider a minimization problem Π and a polynomial-time algorithm $\mathcal{A}$ for solving this problem
- $OPT(\mathcal{I})$: the objective value of an optimal solution for the instance $\mathcal{I}$ of the problem Π
- $SOL(\mathcal{I})$: the objective value of the solution of our algorithm $\mathcal{A}$ for the instance $\mathcal{I}$

approximation ratio

$$SOL(\mathcal{I}) \leq \rho \cdot OPT(\mathcal{I})$$

- $\rho > 1$
- Note that an approximation is as good as ρ is close to 1.

PTAS: going further

The notion of approximation can be refined to target the ratio $1 + \epsilon$.

- We are looking for a family of algorithms parametrized by ϵ.

PTAS: going further

The notion of approximation can be refined to target the ratio $1 + \epsilon$.

- We are looking for a family of algorithms parametrized by ϵ.

Polynomial Time Approximation Scheme

$$SOL(\mathcal{I}) \leq (1 + \epsilon) \cdot OPT(\mathcal{I})$$

with running time polynomial in $|\mathcal{I}|$

- Typically in $\mathcal{O}(|\mathcal{I}|^{\frac{1}{\epsilon}})$
- Here, ϵ is given, thus, the running time is polynomial...
- We can obtain specific algorithms for some values of ϵ, like $\frac{1}{2}$ or $\frac{1}{3}$

Solving the problem

Take 5 minutes to think at an heuristic.

Solving the problem

Take 5 minutes to think at an heuristic.

Think on the way to operate...

Some ideas:

- proceed bin after bin
- minimum space left in the filled bins
- put the items ASAP
- etc.

Solving the problem: Heuristics

- **Next Fit**
  1. Place each item in a single bin until an item does not fit in
  2. When an item don't fit, close it and open a new one
- **Best Fit**
  1. Try to place an item in the fullest bin that can accomodate it
  2. If there is no such bin, open a new one
- **First Fit**
  1. Try to place an item in the first bin that accomodates it
  2. If no such bin is found, open a new one
- **FFD (First Fit Decreasing)**
  Same as FF after sorting the items by decreasing order

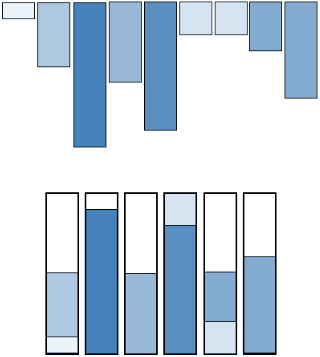
Next Fit

- **Next Fit**
  1. Place each item in a single bin until an item does not fit in
  2. When an item don't fit, close it and open a new one

Methodology

Example followed by the analysis.

Next Fit (example)

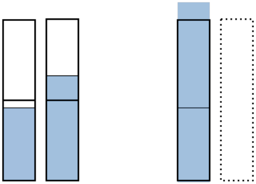


Next Fit (analysis)

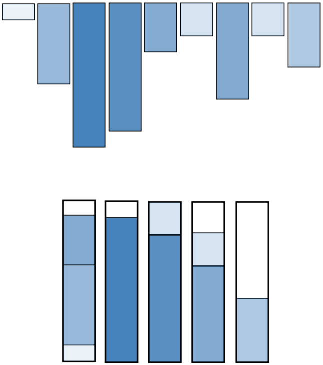
- The argument is that two consecutive bins are filled strictly more than C

Next Fit (analysis)

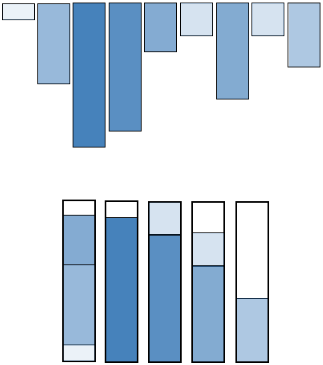
- The argument is that two consecutive bins are filled strictly more than C



Best Fit (Example)

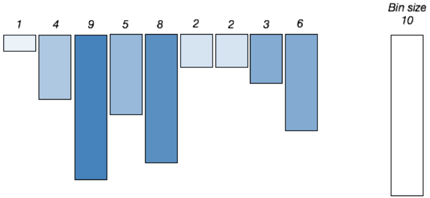


Best Fit (Example)

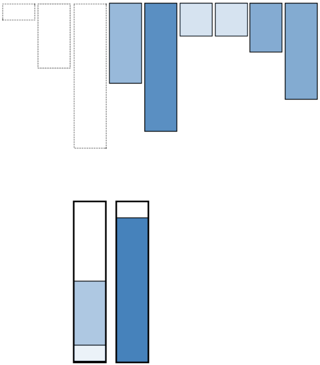


The analysis is let as an exercise.
Another option is to consider WorstFit.

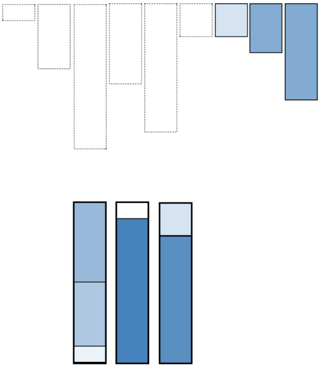
FF example



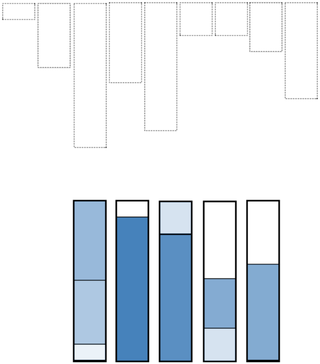
FF example



FF example



FF example

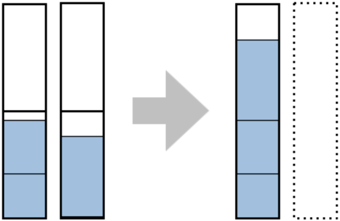


FF informal analysis

- The informal argument is that it is impossible to have two consecutive bins filled less than C.

FF informal analysis

- The informal argument is that it is impossible to have two consecutive bins filled less than C.
- Pictorial proof by contradiction:



FF formal analysis

Proposition

FF is a 2-approximation algorithm.

FF formal analysis

Proposition

FF is a 2-approximation algorithm.

- Assume FF uses m bins (that corresponds to SOL_{FF} in the optimization version of the problem).
- At least (m-1) bins are more than half-full.
- $OPT \geq \Sigma s(a) > \frac{m-1}{2}$
- $2 \cdot OPT > m - 1$ and since OPT and m are integers, $2 \cdot OPT \geq m$

FF formal analysis

Proposition

FF is a 2-approximation algorithm.

- Assume FF uses m bins (that corresponds to SOL_{FF} in the optimization version of the problem).
- At least (m-1) bins are more than half-full.
- $OPT \geq \Sigma s(a) > \frac{m-1}{2}$
- $2 \cdot OPT > m - 1$ and since OPT and m are integers, $2 \cdot OPT \geq m$

Can we do (or even expect) better?

FF formal analysis

Proposition

FF is a 2-approximation algorithm.

- Assume FF uses m bins (that corresponds to SOL_{FF} in the optimization version of the problem).
- At least (m-1) bins are more than half-full.
- $OPT \geq \Sigma s(a) > \frac{m-1}{2}$
- $2 \cdot OPT > m - 1$ and since OPT and m are integers, $2 \cdot OPT \geq m$

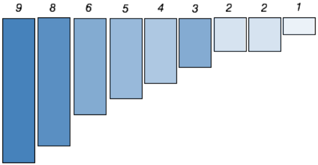
Can we do (or even expect) better?

YES!

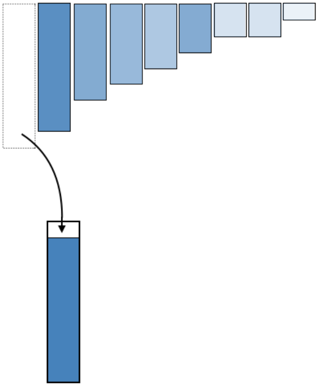
- A refined analysis shows: $SOL_{FF} \leq \frac{17}{10} OPT$ (similarly for BestFit).
- A natural question is to look at other algorithms...

FFD example

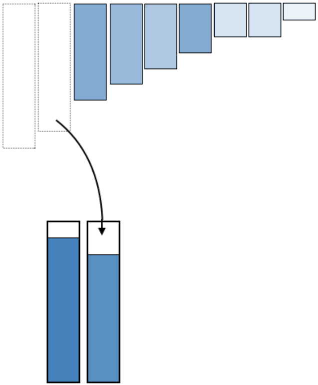
First, sort the items.



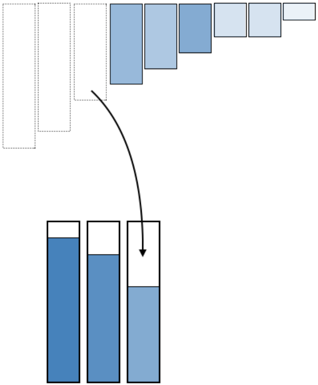
FFD example



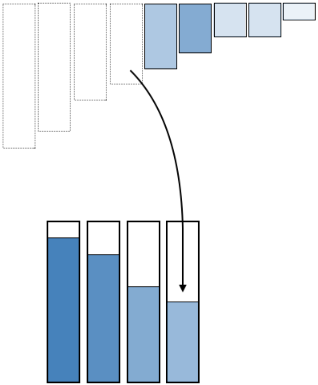
FFD example



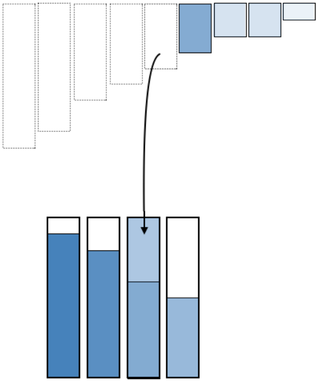
FFD example



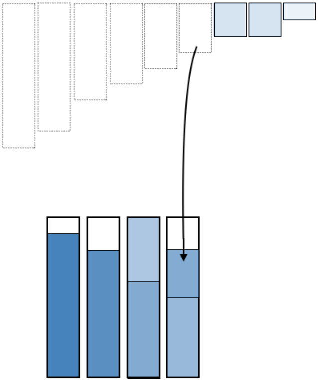
FFD example



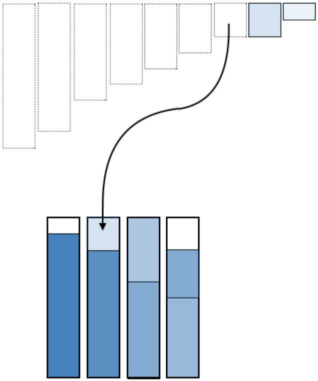
FFD example



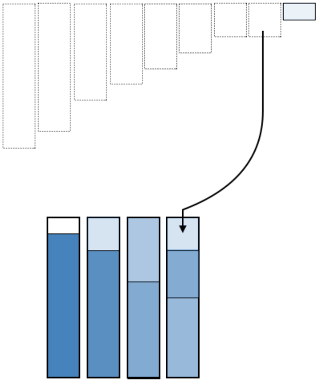
FFD example



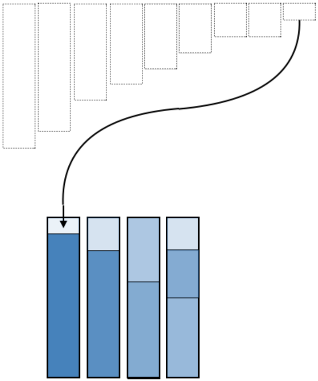
FFD example



FFD example



FFD example



Remark: in this example the bins are all full, but it is not always the case!

FFD analysis

We can show (but it is difficult) that:

- $SOL_{FFD} \leq \frac{11}{9}OPT + \frac{6}{9}$
- It is also possible to show that this bound is tight.

Does it contradict the inapproximation bound?

FFD analysis

We can show (but it is difficult) that:

- $SOL_{FFD} \leq \frac{11}{9} OPT + \frac{6}{9}$
- It is also possible to show that this bound is tight.

Does it contradict the inapproximation bound?
NO!

The result was established for the case $OPT = 2$

$2 \cdot \frac{11}{9} + \frac{6}{9} = 3 + \frac{1}{9}$ bins
that shows that FFD is a $\frac{3}{2}$-approximation

For large values of n, we can obtain much better approximation ratio (asymptotically)

Transforming FF (or NF) in a PTAS

Let analyze two particular cases of the problem.

- δ is given
  Consider that all the item sizes are smaller than δ
- q is given
  Consider that there are only q different sizes

1. FF with limited item sizes

We refine the approximation ratio of FF.

- δ is given

Claim 1

$FF \leq (1+2\delta)OPT + 1$

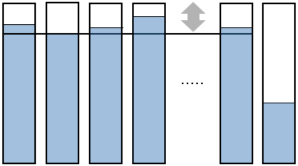
Proof

- if $\delta \geq \frac{1}{2}$ the result is immediate:
  $FF \leq 2 \cdot OPT + 1 \leq (1+2\delta)OPT + 1$

Thus, assume $\delta < \frac{1}{2}$

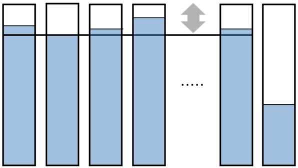
FF with limited item sizes (cont'd)

- $\delta < \frac{1}{2}$
- $(FF - 1)(1 - \delta) \leq OPT$
  Geometric argument.



FF with limited item sizes (cont'd)

- $\delta < \frac{1}{2}$
- $(FF - 1)(1 - \delta) \leq OPT$
  Geometric argument.



- Thus, $FF \leq (1 + 2\delta)OPT + 1$

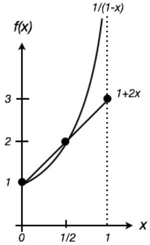
Can you say WHY?

FF with limited item sizes (detail)

- $(FF - 1)(1 - \delta) \leq OPT$
  $FF \leq \frac{1}{1-\delta} OPT + 1$
- $\frac{1}{1-\delta} \leq 1 + 2\delta$ for $\delta < \frac{1}{2}$

FF with limited item sizes (detail)

- $(FF - 1)(1 - \delta) \leq OPT$
  $FF \leq \frac{1}{1-\delta} OPT + 1$
- $\frac{1}{1-\delta} \leq 1 + 2\delta$ for $\delta < \frac{1}{2}$



$$FF \leq (1 + 2\delta) OPT + 1$$

2. FF with limited number of sizes

- q is given

Claim 2

the optimal number of bins can be found in time $\mathcal{O}(n^{2q+1})$
where n is the total number of items.

Proof

- Sort the items by size $S_1, S_2, \ldots, S_q$
  where $|S_j| = n_j$ and $\Sigma_{1 \leq j \leq q} n_j = n$
- Let enumerate the number of possible combinations into a subset S_j:
  it is bounded by $\mathcal{O}(n^q)$
- Compute the optimal number of bins for each subset by Dynamic Programming.
  There are at most OPT steps, each costs less than $\mathcal{O}(n^{2q})$
  and obviously, $OPT \leq n$

Detail for computing the number of possible subsets

- The idea here is to use a smart encoding.

Example

let consider the following instance:

- $(2, 4, 9, 3, 7, 9, 3, 2, 3)$
- The corresponding multi-set ($q = 5$) is
- $(2, 3, 4, 7, 9)$
- We represent the encoding by a vector of dimension q as follows:
- $(2, 3, 7, 3) \rightarrow (1, 2, 0, 1, 0)$

This way, the number of subsets is in $\mathcal{O}(n^q)$ and not $\Theta(2^n)$

Combining both cases together...

- Consider an instance $\mathcal{I}$ of the general problem and ϵ
  If all the item sizes are $\leq \frac{\epsilon}{2}$ then by Claim 1:
  $FF\ (\mathcal{I}) \leq (1+\epsilon)OPT\ (\mathcal{I})\ +1$

- Assume that all the item sizes are $> \frac{\epsilon}{2}$
  By Claim 2, there exists a packing algorithm –call it A($\mathcal{I}$)– for which:
  $SOL_A\ (\mathcal{I}) \leq (1+\epsilon)OPT\ (\mathcal{I})\ +1$

Construction of the combined approximation algorithm

Let introduce a parameter x whose value will be determined later.

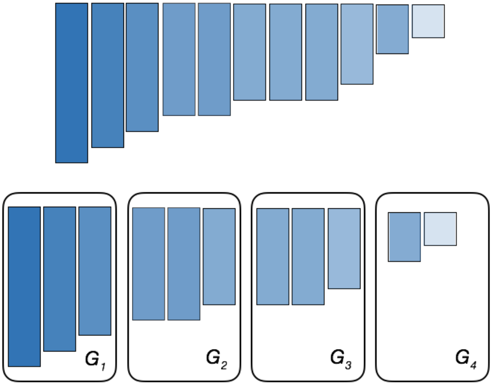
- Sort $\mathcal{I}$ in non-decreasing order.
- Split it into $\lceil \frac{n}{x} \rceil$ groups of x elements each.
  Denote $\lceil \frac{n}{x} \rceil$ in short by n_x and the i-th group by G_i
- Pack G_1 in at most x bins.
- Change all the sizes in G_i (for $i \geq 2$) to the largest size in this group.
  Call this new instance $\mathcal{I}$'.
- Determine the optimal packing of $\mathcal{I}$' by using the process of Claim 2.

Proposition

$OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

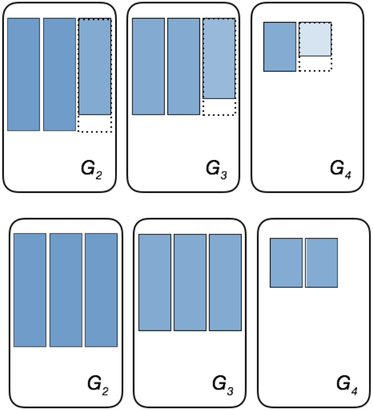
Example

- Let us consider the following instance $\mathcal{I}$



Example (cont'd)

- The transformed instance $\mathcal{I}'$

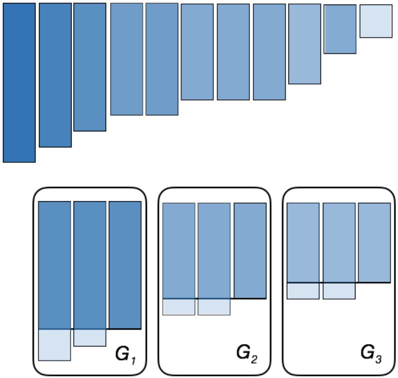


Another rounding

- Proof of:
  $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

- Consider the new derived instance $\mathcal{I}''$ constructed as follows:
- Remove the smallest group (G_{n_x})
- Change every element in group G_i for $1 \leq i \leq n_x - 1$ to the smallest item in this group.

Example instance $\mathcal{I}''$



Analysis of the rounded algorithms

- Proof of:
  $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

- Clearly, $OPT(\mathcal{I}'') \leq OPT(\mathcal{I})$
  WHY?

Analysis of the rounded algorithms

- Proof of:
  $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

- Clearly, $OPT(\mathcal{I}'') \leq OPT(\mathcal{I})$
  WHY?
- Since $\mathcal{I}''$ has less elements and they are smaller.

Analysis of the rounded algorithms

- Proof of:
  $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

- Clearly, $OPT(\mathcal{I}'') \leq OPT(\mathcal{I})$
  WHY?
- Since $\mathcal{I}''$ has less elements and they are smaller.

- $OPT(\mathcal{I}') \leq OPT(\mathcal{I}'')$
  WHY?

Analysis of the rounded algorithms

- Proof of:
  $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

- Clearly, $OPT(\mathcal{I}'') \leq OPT(\mathcal{I})$
  WHY?
- Since $\mathcal{I}''$ has less elements and they are smaller.

- $OPT(\mathcal{I}') \leq OPT(\mathcal{I}'')$
  WHY?
- 1. $\mathcal{I}'$ may have less elements:
     $|G_{n_x}| \leq |G_1|$ and all the other sets have the same size in both.
  2. $\mathcal{I}'$ has a smaller total sum
     since max $G_{i+1} \leq$ min G_i for $1 \leq n_x - 1$

$OPT(\mathcal{I}') \leq OPT(\mathcal{I}'') \leq OPT(\mathcal{I})$

We are almost at home!

- the running time of the algorithm is in $\mathcal{O}(n^{2n_x+1})$
- its approximation ratio is:
  $SOL_A(\mathcal{I}) \leq OPT(\mathcal{I}) + x$
- We can then choose x
  $x = \lceil \epsilon \cdot \Sigma_{1 \leq i \leq n} s_i \rceil$
  WHY?

We are almost at home!

- the running time of the algorithm is in $\mathcal{O}(n^{2n_x+1})$
- its approximation ratio is:
  $SOL_A(\mathcal{I}) \leq OPT(\mathcal{I}) + x$
- We can then choose x
  $x = \lceil \epsilon \cdot \Sigma_{1 \leq i \leq n} s_i \rceil$
  WHY?
  Because $\lceil \epsilon \cdot \Sigma_{1 \leq i \leq n} s_i \rceil \leq \epsilon \cdot OPT + 1$

Synthesis of the asymptotic-PTAS: the algorithm

1. we are given $\epsilon < 1$
2. split the instance into two parts of small $\leq \frac{\epsilon}{2}$ and large $> \frac{\epsilon}{2}$
3. round the large items of $\mathcal{I}'$ with $q = \lceil \epsilon \cdot \Sigma_{a \in \mathcal{I}'} s(a) \rceil$
4. determine the optimal packing of this rounded sub-instance
5. keep the same packing with the original values
6. pack the remaining small items using FirstFit

Notice that packing the small items can done without increasing the ratio since there is an area at most $\frac{\epsilon}{2}$ left in each bin, may be we will need one extra bin that will be only partially filled.

Final result

Asymptotic PTAS

- The running time expressed in ϵ is:
  $\mathcal{O}(n^{\frac{4}{\epsilon^2}+1})$
  It is polynomial because ϵ is fixed.
- The approximation ratio is $(1+\epsilon)OPT+1$

Final remark: How to beat FFD?

- For beating FFD, we need to choose $\epsilon \leq \frac{2}{9}$
- As a consequence, the APTAS will have the following running time: $\mathcal{O}(n^{4/(2/9)^2+1}) = \mathcal{O}(n^{82})$
- more than the number of atoms in the universe...