REOP - Class 8 : Bin packing & Facility location

I/ Bin packing

Putting objects in containers without exceeding weight limit

A) One container, maximize total object value → Knapsack

Knapsack : n objects weights w_i values c_i
Find $S \subseteq [n]$ so that $\sum_{i \in S} w_i \leq W$ & $\sum_{i \in S} c_i$ is maximal

Last session : dynamic programming algo with complexity $O(nW)$
pseudo polynomial since it depends polynomially on the numerical value W of one parameter $= O(n\, 2^{\log_2 W})$
& not just on its encoding size $\log_2 W$

This is an "exponential" algorithm but
* there could be faster algorithms
* we didn't prove NP-hardness by finding a reduction

<u>Thm</u> : The knapsack pb is NP-hard

1) ILP formulation

Decision variable: $x_i = \begin{cases} 1 & \text{if we select object } i \text{ in } S \\ 0 & \text{otherwise} \end{cases}$

Objective : $\sum_{i \in S} c_i = \sum_{i=1}^{n} x_i c_i$

Constraints : $\sum_{i \in S} w_i \leq W \iff \sum_{i=1}^{n} x_i w_i \leq W$

(ILP) $\max_x \sum_{i=1}^{n} x_i c_i$ s.t. $\begin{cases} \sum_{i=1}^{n} x_i w_i \leq W \\ \forall i,\ x_i \in \{0,1\} \end{cases}$

linear / continuous relaxation

(LP) $\max_x \sum_{i=1}^{n} x_i c_i$ s.t. $\begin{cases} \sum x_i w_i \leq W \\ \forall i,\ x_i \in [0,1] \end{cases}$

useful for
- upper bound
- Branch & Bound

13.1.2 Q3 : We don't actually need the simplex to solve (LP), we have a fast direct method

2) Solving the LP relaxation

Hyp: there is no object i with $w_i > W$

We sort the objects by decreasing "utility" $\frac{c_i}{w_i}$ (price per kg)

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \cdots \geq \frac{c_n}{w_n}$$

Let j be the largest index such that $\sum_{i \leq j} w_i \leq W$ (reordered)

- We can take the first j items
- We must split item $j+1$

We define $x \in [0,1]^n$ by

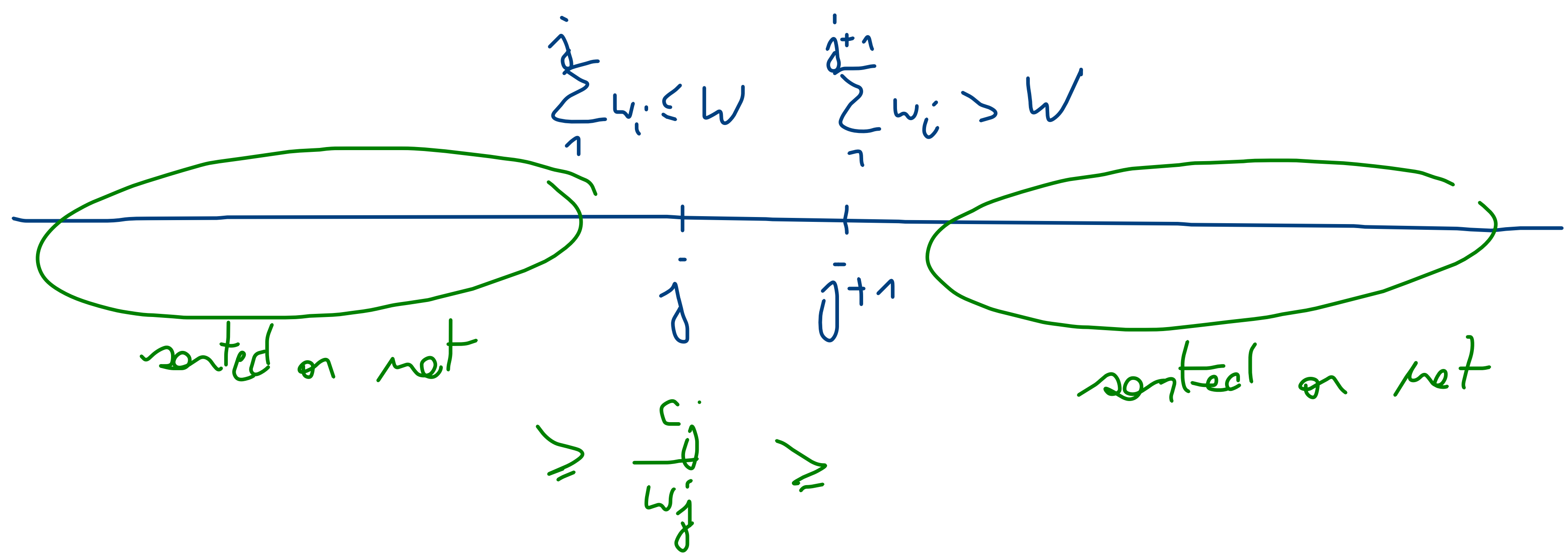
$$\begin{cases} x_i = 1 & \text{if } i \leq j \\ x_i = 0 & \text{if } i > j+1 \\ x_{j+1} = \dfrac{W - \sum_{i \leq j} w_i}{w_{j+1}} & \end{cases}$$

We can check easily that $\sum x_i w_i = W$ $\therefore$ x_i feasible for (LP)

Thm: x is optimal for LP

Proof: exchange argument

The linear relaxation can be solved in $O(n \log n)$
Actually we don't need to sort the list: ↳ sorting
we only need a weighted median



This can be found in $O(n)$

Remark: This sorting method only works because we can split the last item in the LP
Otherwise it may be more interesting to remove an object and add two others: doesn't work for the ILP

3) Constructive heuristic & approximation algorithm

Let x be the solution to (LP) constructed above: how to deduce an approximation algorithm?

Naive construction : let $y_i = \begin{cases} 1 & \text{if } x_i = 1 \\ 0 & \text{if } x_i \in [0, 1[\end{cases}$ $(1 \leq i \leq j)$

leave out the last item

We have y feasible for the (ILP) but no performance guarantee

Second integer solution : $z_i = \begin{cases} 1 & \text{if } x_i \in]0, 1[\\ 0 & \text{otherwise} \end{cases}$ $(i = j+1)$

z is also feasible for the (ILP)

fractional

$$\text{cost}(y) + \text{cost}(z) = \sum_{i=1}^{j+1} c_i \geq \sum_{i=1}^{j} c_i + x_{j+1} c_{j+1}$$

$$= \text{val}(LP)$$

$$\geq \text{val}(ILP)$$

At least one of y & z has cost $\geq \frac{1}{2}$ val(ILP)

Alg : 1) Construct y & z
2) Select the best

is a $\frac{1}{2}$-approximation algorithm : its output is at least $\frac{1}{2}$ as good as the optimum of the knapsack problem

B) Several containers, minimize nb of containers used → bin packing

We have n items with size a_i, max container size is W

Find an integer k and an assignment $\sigma : \underset{\text{items}}{[n]} \longrightarrow \underset{\text{containers}}{[k]}$

such that for each container $j \in [k]$, $\sum_{\substack{i \in [n] \\ \sigma(i) = j}} a_i \leq W$ and k is minimal

sum over j

Thm: Bin packing is NP-hard

Easy upper bound: $n \geq k^{opt}$

lower bound: $\left\lceil \frac{\sum a_i}{W} \right\rceil \leq k^{opt}$ because $\sum_{i=1}^{n} a_i \leq k^{opt} W$

(the smallest integer $\geq \frac{\sum a_i}{W}$)

1) Heuristics

NEXT-FIT : Take items one after the other, in any order and when an item doesn't fit in the current box, close it & open a new box

B.2.2 Q7 : This is a 2-approximation algorithm

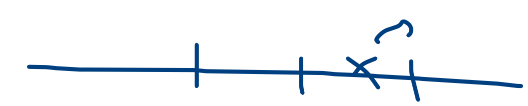
For $j \leq \left\lfloor \frac{k}{2} \right\rfloor$, what can we say of $\displaystyle\sum_{i:\sigma(i)\in\{2j-1,\,2j\}} a_i$?

$$\sum_{i:\sigma(i)\in\{2j-1,2j\}} a_i > W,$$ otherwise we wouldn't have opened $2j$ because it would have fit inside $2j-1$

sum over j

$$\sum_{i=1}^{n} a_i > W\left\lfloor \frac{k}{2} \right\rfloor \iff \frac{\sum a_i}{W} > \left\lfloor \frac{k}{2} \right\rfloor$$

$\rightarrow$ integer



$$\implies \left\lceil \frac{\sum a_i}{W} \right\rceil - 1 \geq \left\lfloor \frac{k}{2} \right\rfloor \geq \frac{k-1}{2}$$

$$\iff k \leq 2\underbrace{\left\lceil \frac{\sum a_i}{W} \right\rceil}_{\leq k^{opt} \text{ as we saw}} - 1 \leq 2k^{opt} - 1$$

FIRST-FIT : Take the items in any order, keep all containers open & put each item in the first container where it fits

FIRST-FIT-DECREASING : This but sort items by decreasing size first

Thm : $k^{NF} \leq 2k^{opt} - 1$

$k^{FF} \leq 17/10\, k^{opt}$

$k^{FFD} \leq 3/2\, k^{opt}$ & $\frac{3}{2}$ is the best ratio that can be obtained in poly time

$k^{FFD} \leq 11/9\, k^{opt} + 2/3$ & this bound is tight

2) ILP formulation

We assume we have K boxes available (worst case: $K = n$)

Decision variables: $z_j = 1$ if box j is used
$y_{ij} = 1$ if item i goes in box j } binary

$\min \sum_{j=1}^{K} z_j$ subject to

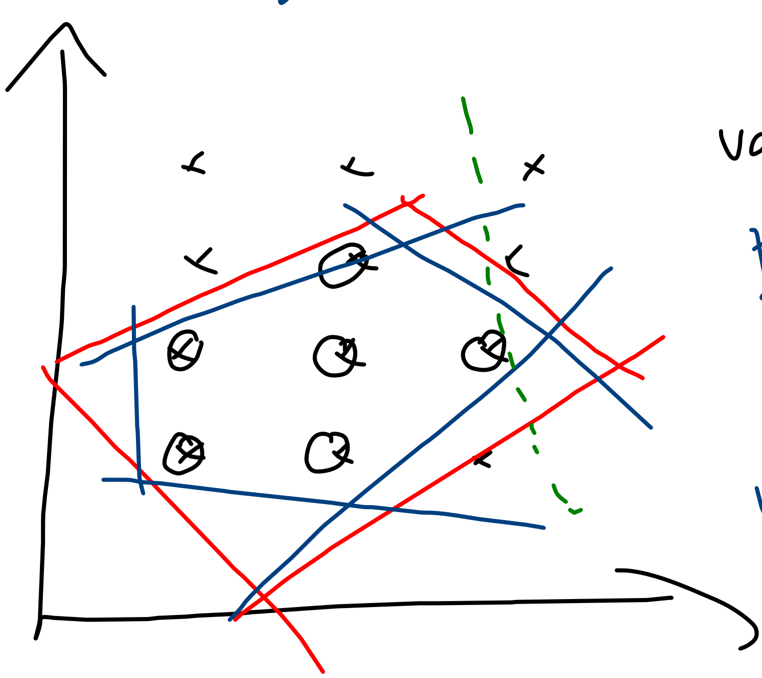
$\forall i, \sum_{j=1}^{K} y_{ij} = 1$ each item into 1 box

$\forall j, \sum a_i y_{ij} \leq W z_j$ box size constraints for used boxes + ensure that no object goes into a non used box

$y \in \{0,1\}^{n \times k}$
$z \in \{0,1\}^{K}$

(ILP)

Branch & Bound works better if the linear relaxations are tight



$\text{val}(\text{ILP}) \geq \text{val}(\text{LP}) \geq \text{val}(\text{LP})$

Blue relaxation is better / tighter

One way to speed up B & B is adding valid inequalities (or cuts)

removes continuous solutions but no integer solutions

Ex 13.3

2. $\sum z_j \geq \left\lceil \frac{\sum a_i}{W} \right\rceil$

1. Why can we add $z_j \geq z_{j+1}$ to (ILP)?

There is always an optimal solution where the boxes used are the first k ones: $z_j = 1$ $j \in [1, k]$ by symmetry
$z_j = 0$ $j \in [k+1, K]$

Removing others = removing symmetric (equivalent solutions)
→ smaller search space in B&B

2. Why can we add $\sum_{j=1}^{K} z_j \geq \left\lceil \frac{\sum a_i}{W} \right\rceil$?

It's a lower bound on the objective

continuous solutions satisfy $\sum z_j \geq \frac{\sum a_i}{W}$

integer solutions $\sum z_j \geq \left\lceil \frac{\sum a_i}{W} \right\rceil$ (stronger)

Valid inequality

II/ Facility location

Set of customers $\mathcal{D}$ that must be served with facilities chosen from a set $\mathcal{F}$

Opening cost f_i $i \in \mathcal{F}$ Serving cost c_{ij} $i \in F$, $j \in \mathcal{D}$

Find a subset $Y \subseteq \mathcal{F}$ of facilities to open & an assignment $\sigma : \mathcal{D} \to Y$ of clients to open facilities such that

$$\sum_{i \in Y} f_i + \sum_j c_{\sigma(j) j} \text{ is minimal}$$

Thm: Facility location is NP-hard

A) ILP formulation

$y_i = 1$ if facility i is opened
$x_{ij} = 1$ if facility i serves client j

(very similar to bin packing)

$$\min \sum_i y_i f_i + \sum_i \sum_j x_{ij} c_{ij}$$

subject to

$$\begin{array}{lll} x_{ij} \leq y_i & \forall i,j & \text{serve clients from open facility} \\ \sum_{i \in F} x_{ij} = 1 & \forall j & \text{1 facility for every client} \\ x_{ij} \in \{0,1\} & \forall i,j & \\ y_i \in \{0,1\} & \forall i & \end{array}$$

B) Local search

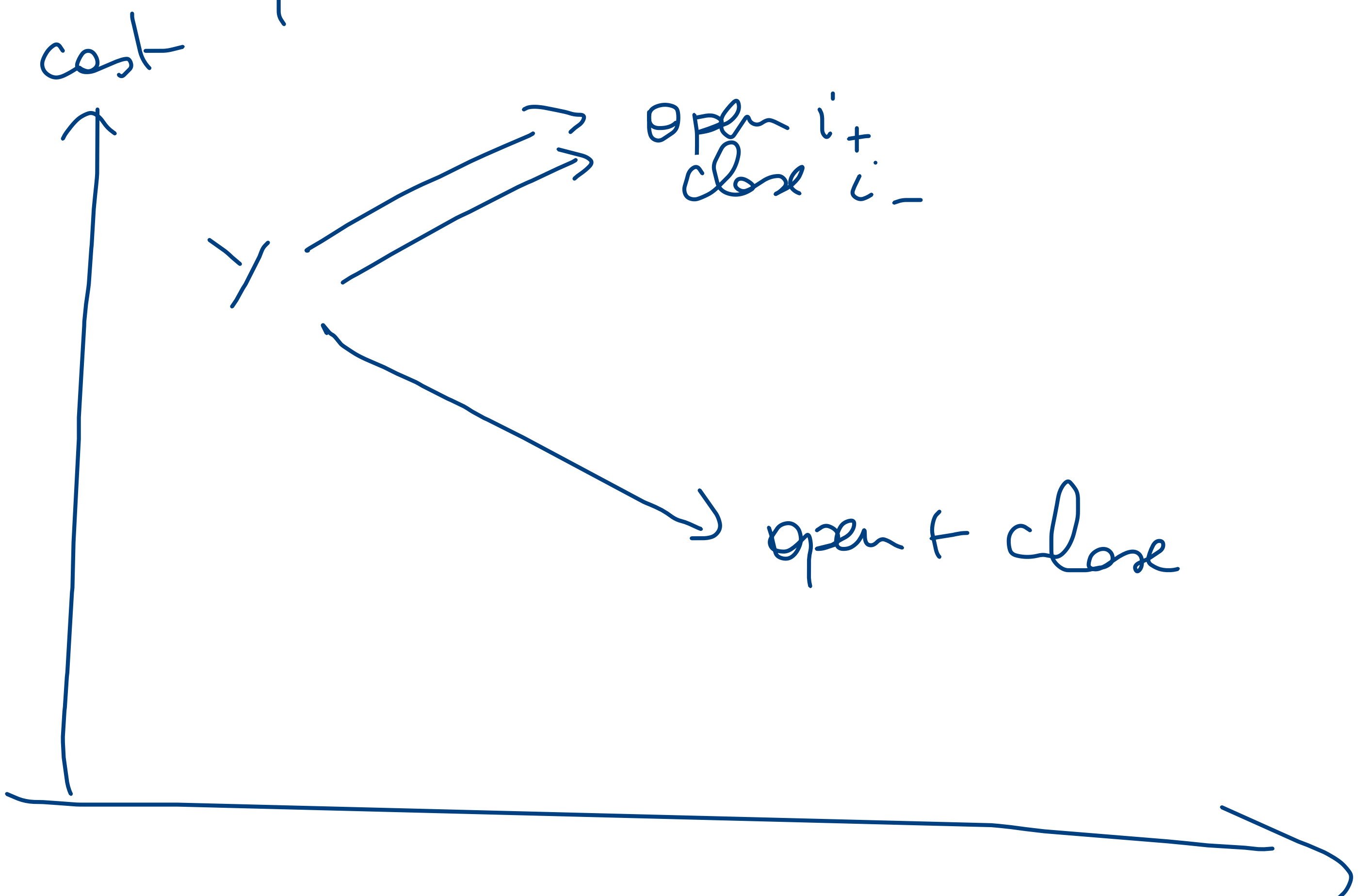
If the set of opened facilities Y is given, the optimal assignment function σ can easily be computed:

$$\sigma(j) = \operatorname{argmin}_{i \in Y} c_{ij}$$

This suggests a 2-step local search: at each iteration
1. modify Y
2. assign clients optimally & compute cost to compare

Possible modifications for Y (opened facilities):

- open new one $\quad Y \to Y \cup \{i\}$
- close $\quad Y \to Y \setminus \{i\}$
- swap $\quad Y \to Y \cup \{i_+\} \setminus \{i_-\}$



$\boxed{\text{HW!}}$ Ex 13.4