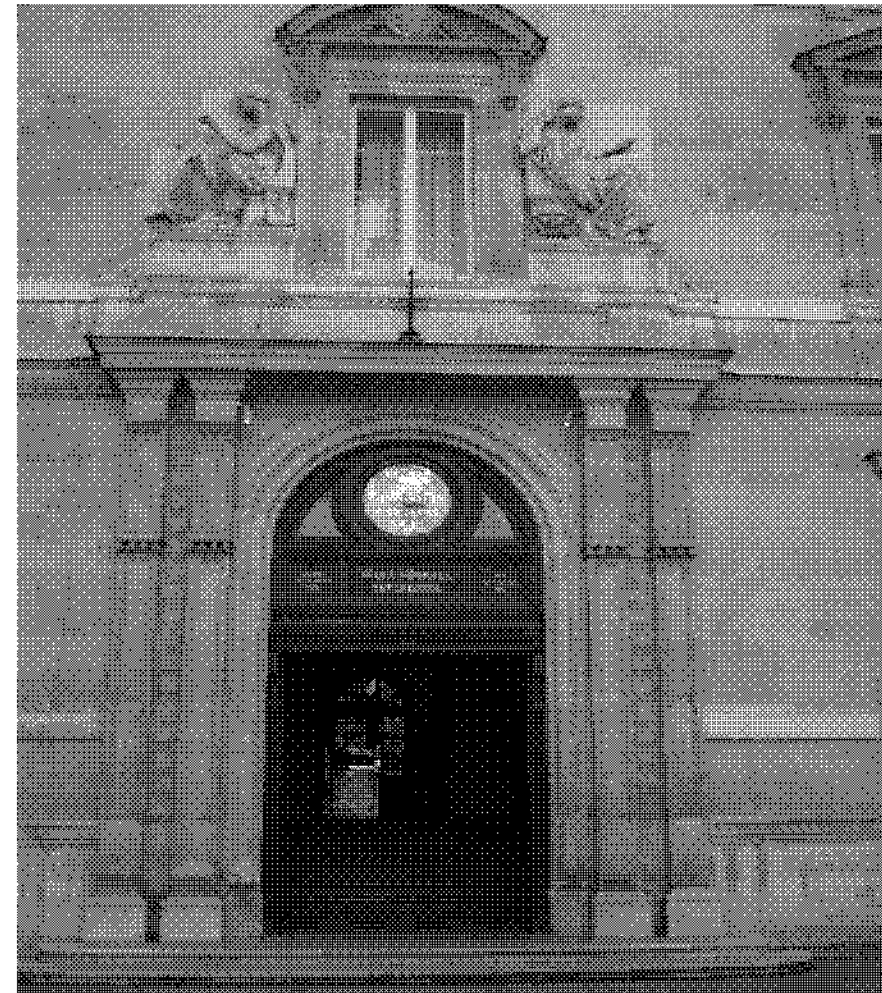


Algorithms for Approximation

Claire Mathieu
CNRS, DI-ENS and Brown

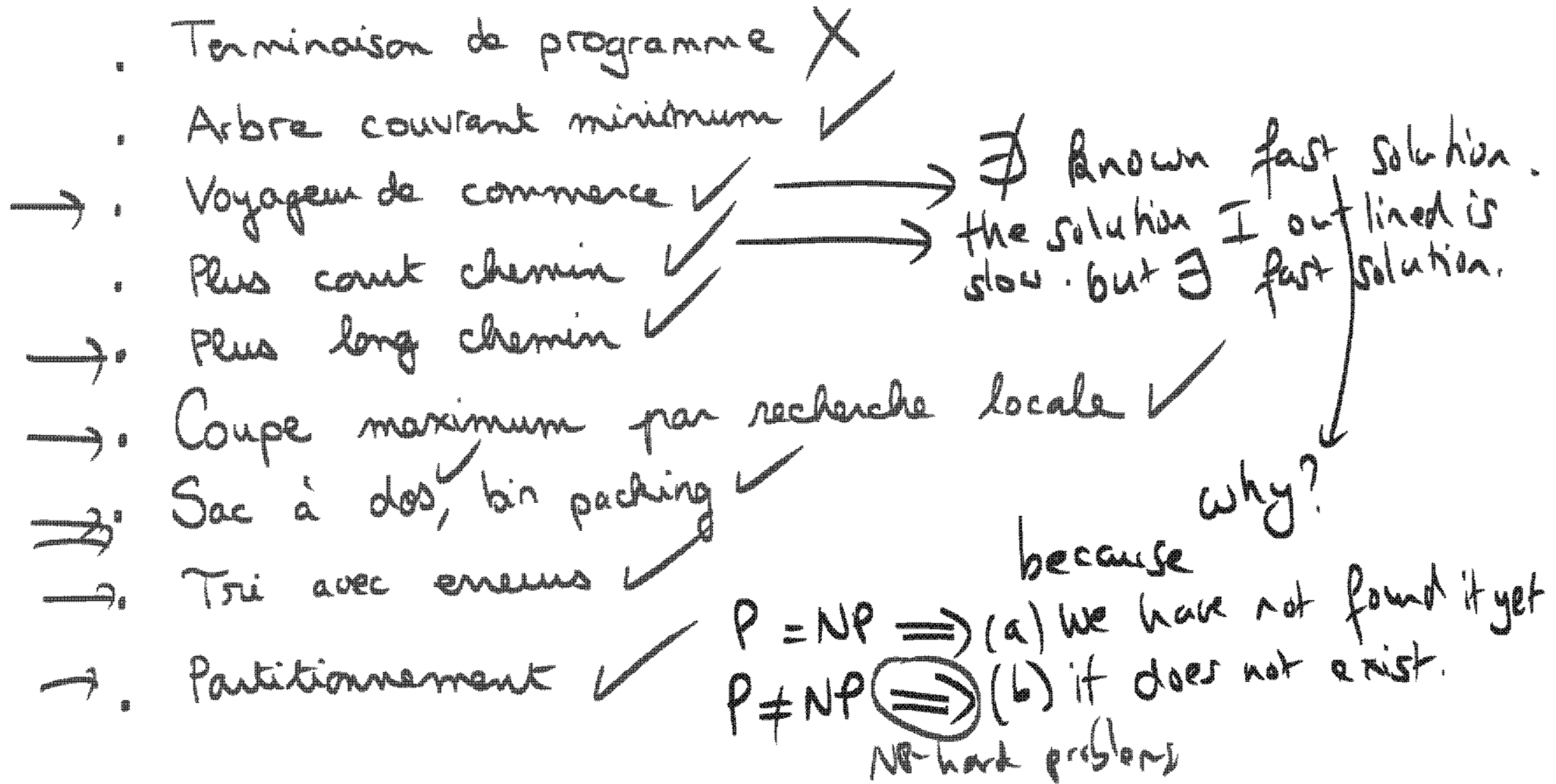


Problème

Modèle

Algorithme

Résolution constructive et efficace de problèmes



Classification of problems:

Minimum spanning tree

Shortest path

2-SAT

} easy

TSP

longest path

2-SAT

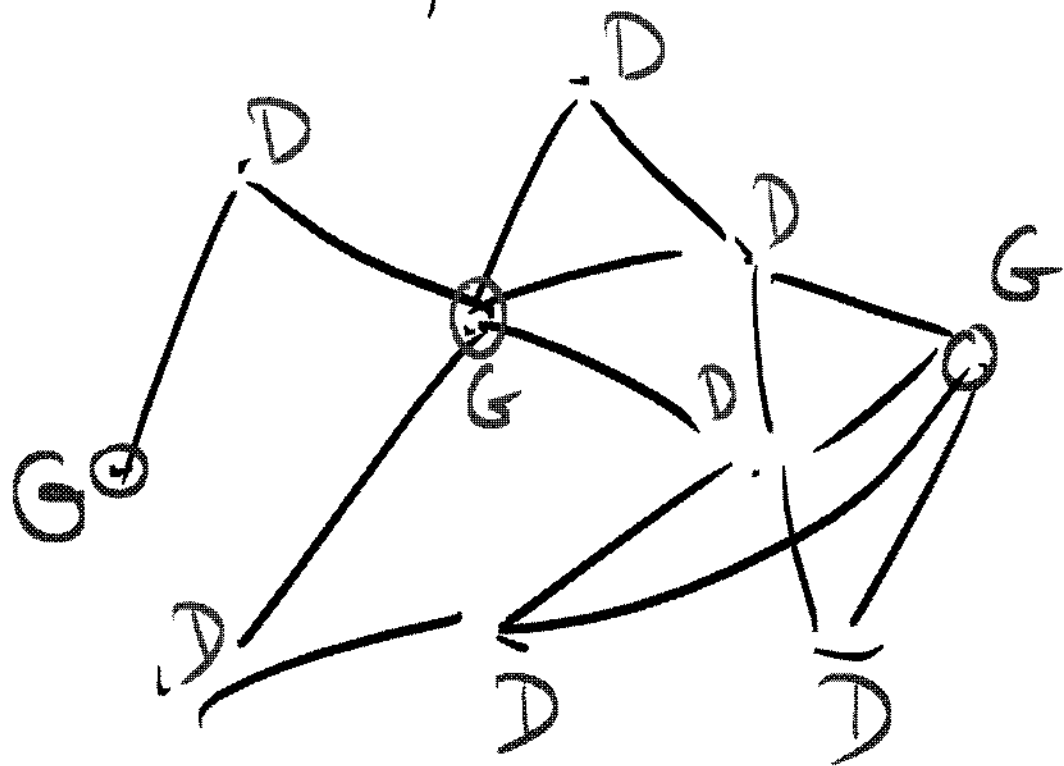
max

} NP-hard

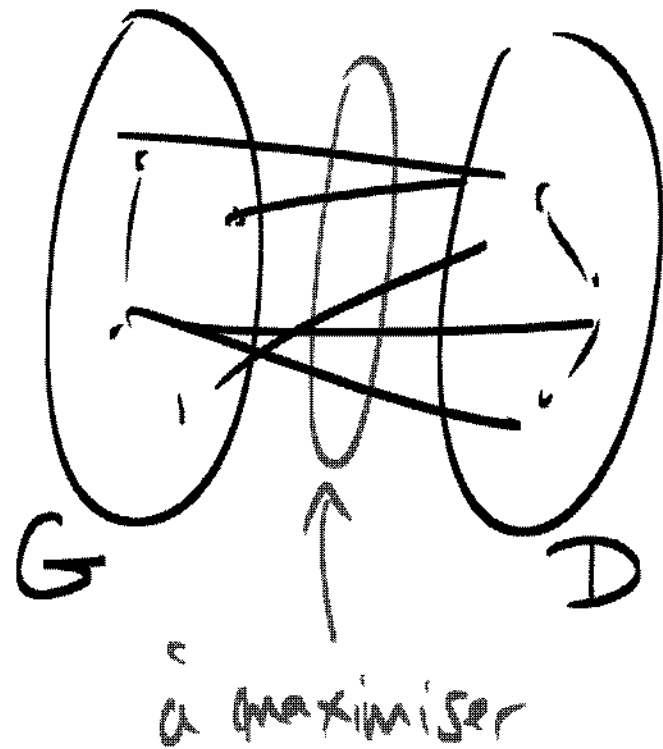
Dealing with NP-hard problems: heuristics, approximation, special cases, average-case analysis

Coupe maximale par recherche locale

graphe $G = (V, E)$



Coupe



NP-difficile.

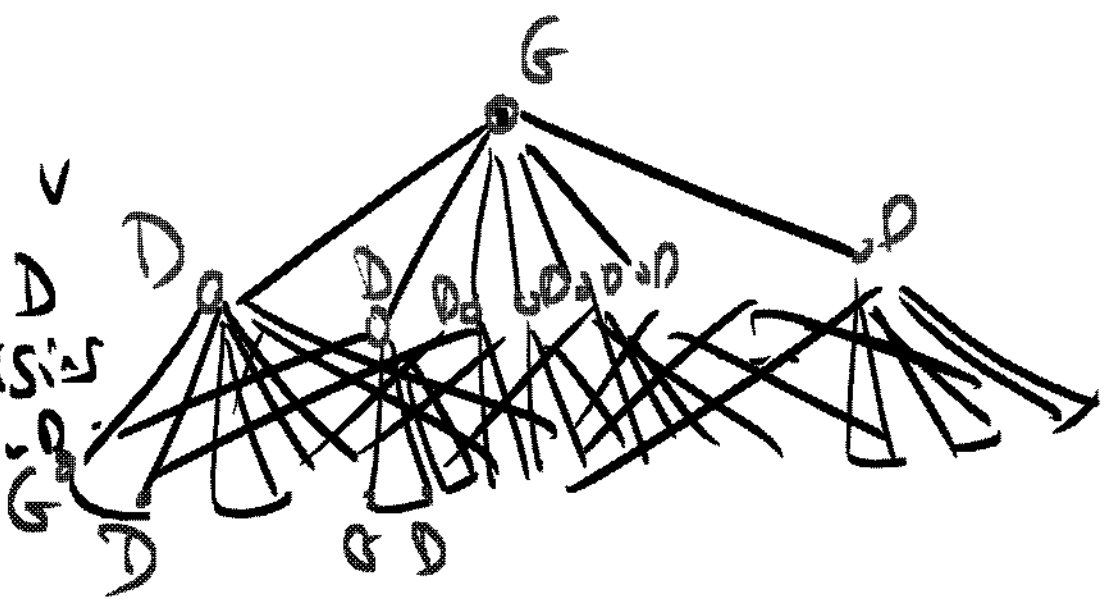
But: $\text{coup}(G, D)$

log.

$\# \text{arêtes in } G \setminus D \geq \alpha \cdot \text{OPT}$
 α : approximation ratio.

Glouton.

Pour chaque sommet v
 placer v de G ou D
 selon le nb de voisins
 de v étiquetés G ou D .



Recherche locale

initialient,

partition arbitraire of $V = G \cup D$
 - tant qu' \exists vertex v tq changer v de côté
 améliore la taille de la coupe, changer v .

Aléatoire.

$\forall v$

placer v $\begin{cases} \text{ds } G & \text{avec proba } 50\% \\ \text{ds } D & \text{" " " } 50\% \end{cases}$

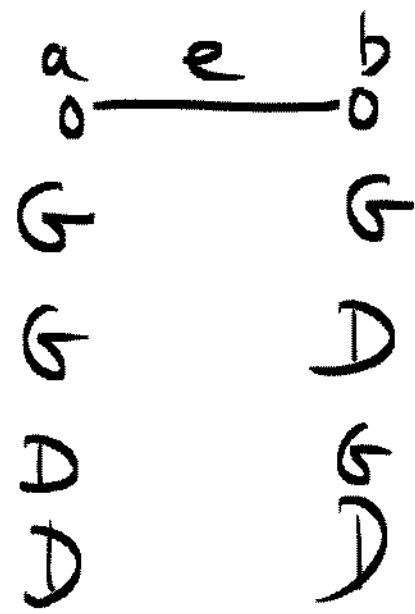
What is α ?

Valeur (coupe) $\geq \frac{1}{2}$ OPT

Pourquoi?

$$\frac{1}{2} \text{OPT} \leq \frac{1}{2} |E|.$$

1/4
1/4
1/4
1/4



e n'est pas ds la coupe
e est
e est
e n'est pas

$$P(e \text{ est ds la coupe}) = 1/2.$$

$$|coupe| = \sum_{e \in E} \mathbb{1}(e \text{ est ds la coupe}).$$

$$E |coupe| = \sum_{e \in E} E(\mathbb{1}(e \text{ est ds la coupe}))$$

$$= \sum_{\text{edges } e} \frac{1}{2} = \frac{|E|}{2}.$$

QED

Improving the ratio.

$$\frac{1}{2} = 0.5$$

1994: Goemans Williamson

0.878...

2001. What defined the "unique games conjecture",
if the UGC holds
then no polynomial time algorithm
with ratio better than 0.878...

Sac à dos / Knapsack.

Bin Packing

The movers' problem.

Input: items $0 < a_1, a_2, \dots, a_n < 1$.

Output: placement of items in bins

Such that

$\{a_i \text{ in same bin}\}$ has total size ≤ 1 .

Goal: minimize # bins used.

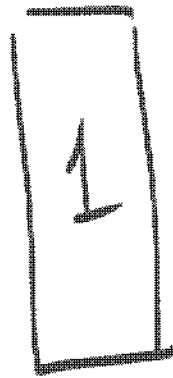


bin-packing is NP-hard.
 (Knapsack is NP-hard)
 idea: rounding.

- ① Round item sizes up so that every size is an integer multiple of 0.25
- ② Solve bin packing for rounded instance
- ③ Hence a packing for initial instance.

Comment: "it doesn't work", b/c if all input items are = 0.0001 but we rounded them to 25%.

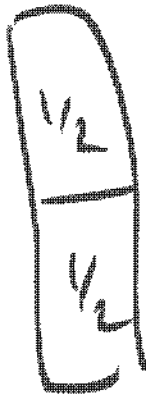
If all objects are in $\{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$.



x_1



x_2



x_3



x_4



x_5

For all $(x_1, \dots, x_5) \in \{0, \dots, n\}^5$
check if input fits.

pick solution s.t.
bins is minimum.

$$\approx n \log n \left(\frac{1}{\epsilon}\right)^8$$

What is the approximation ratio α ?

output bins $\leq \alpha \cdot \text{OPT}$

I want $\alpha = 1.01$.

Comment: "it doesn't work", b/c if all input items are $= 0.0001$ but we rounded them to 25%.

Answer: ignore the really small items.

Comment: "it still doesn't work", b/c:

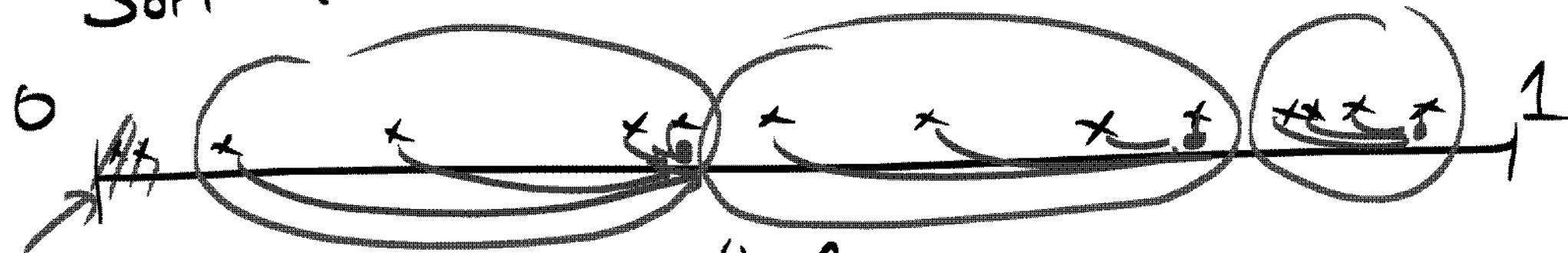
input: $\frac{1}{2} + 0.00001, \dots, \frac{1}{2} + 0.00001$
 $\frac{1}{2} - 0.00001, \dots, \frac{1}{2} - 0.00001$.

before rounding: $\frac{n}{2}$ bins suffice.

after rounding: $\frac{3n}{4}$ bins are necessary.

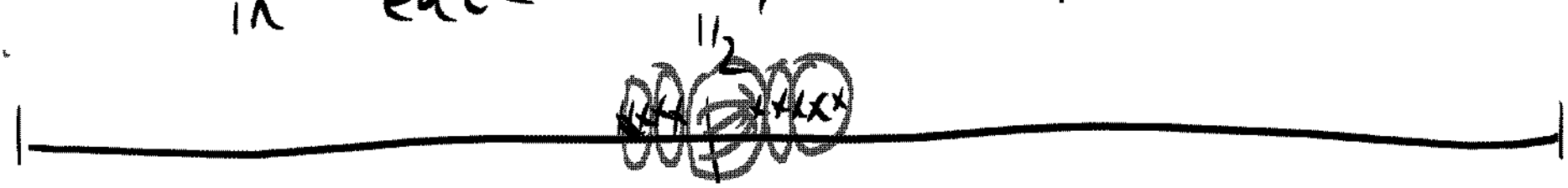
Answer (de la Vega Lueker 1981): To get 1.01,
adaptive rounding. use blocks of size $(0.01)^2 n$

Sort items



Small items are ignored (< 0.01)

partition into blocks. in each block, round up to the same value.



Thm: $\forall \epsilon > 0$ there is a polynomial-time approximation algorithm with ratio $\leq 1 + \epsilon$.

Minimum feedback arc set

Input: tournament: complete directed graph.

For every pair of vertices i, j ,

either (i, j) is in E  means: $i < j$

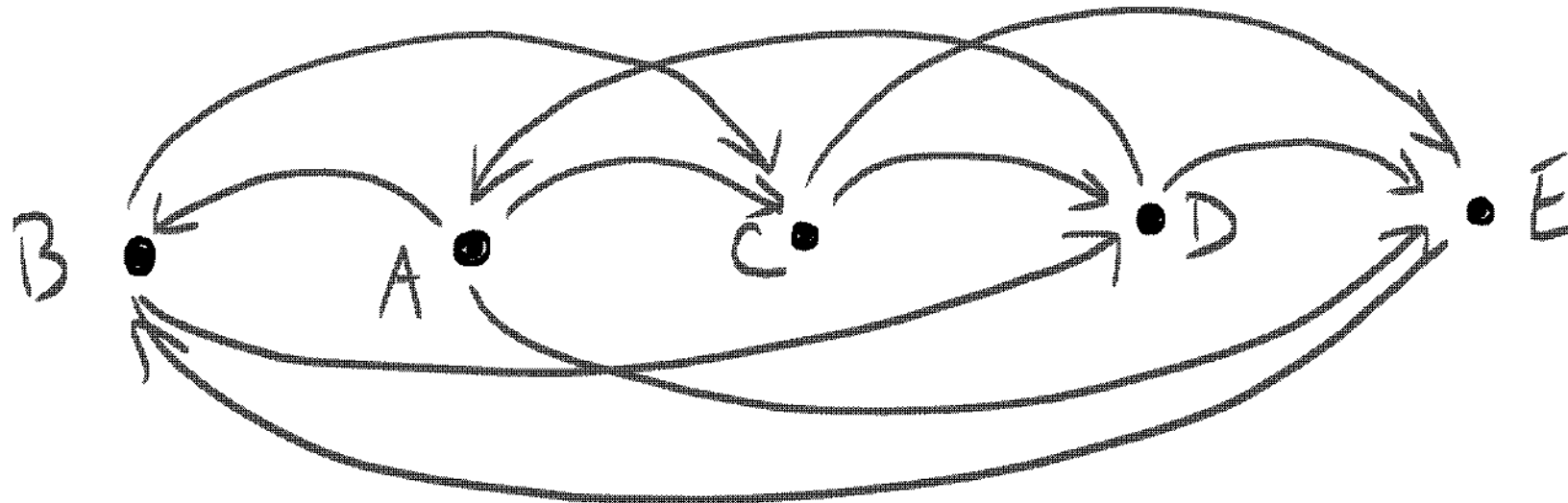
or (j, i) is in E  means: $j < i$

but not both

Output:

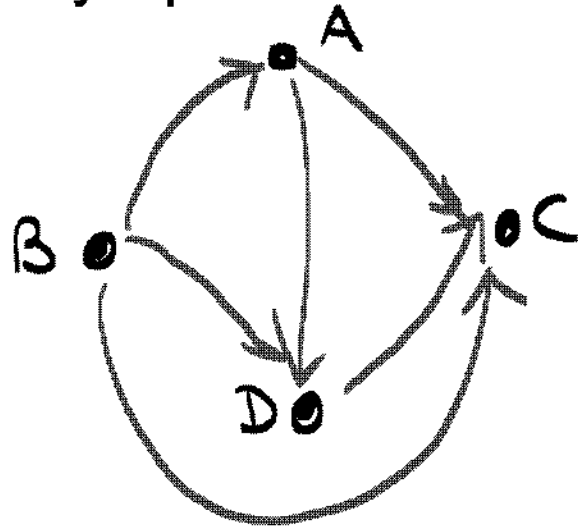
Ordering of vertices "according to the tournament comparisons"

Objective: $\max \sum_{(i,j) \in E} \mathbb{1}(i \text{ precedes } j \text{ in ordering})$



Theorem [Alon et al. 2007]: The minimum feedback arc set on tournaments is NP-hard.

Easy special case: transitive tournaments



if (i,j) and (j,k) are in E
then (i,k) is in E

For perfect input = transitive tournaments,
the problem is easy: sorting

Algorithm for perfect inputs

Pick a vertex i

Let $L = \{\text{vertices } j \text{ such that } (j,i) \text{ is in } E\}$

and $R = \{\text{vertices } j \text{ such that } (i,j) \text{ is in } E\}$

Recursively solve the problem for L and for R

Output: (ordering of L , i , ordering of R)

What about imperfect inputs?

better objective: $\min \sum_{(i,j) \in E} \mathbb{1}(j \text{ precedes } i \text{ in ordering})$

perfect input: value = 0

Randomized Quicksort Algorithm

Pick a random vertex i

Let $L = \{\text{vertices } j \text{ such that } (j,i) \text{ is in } E\}$

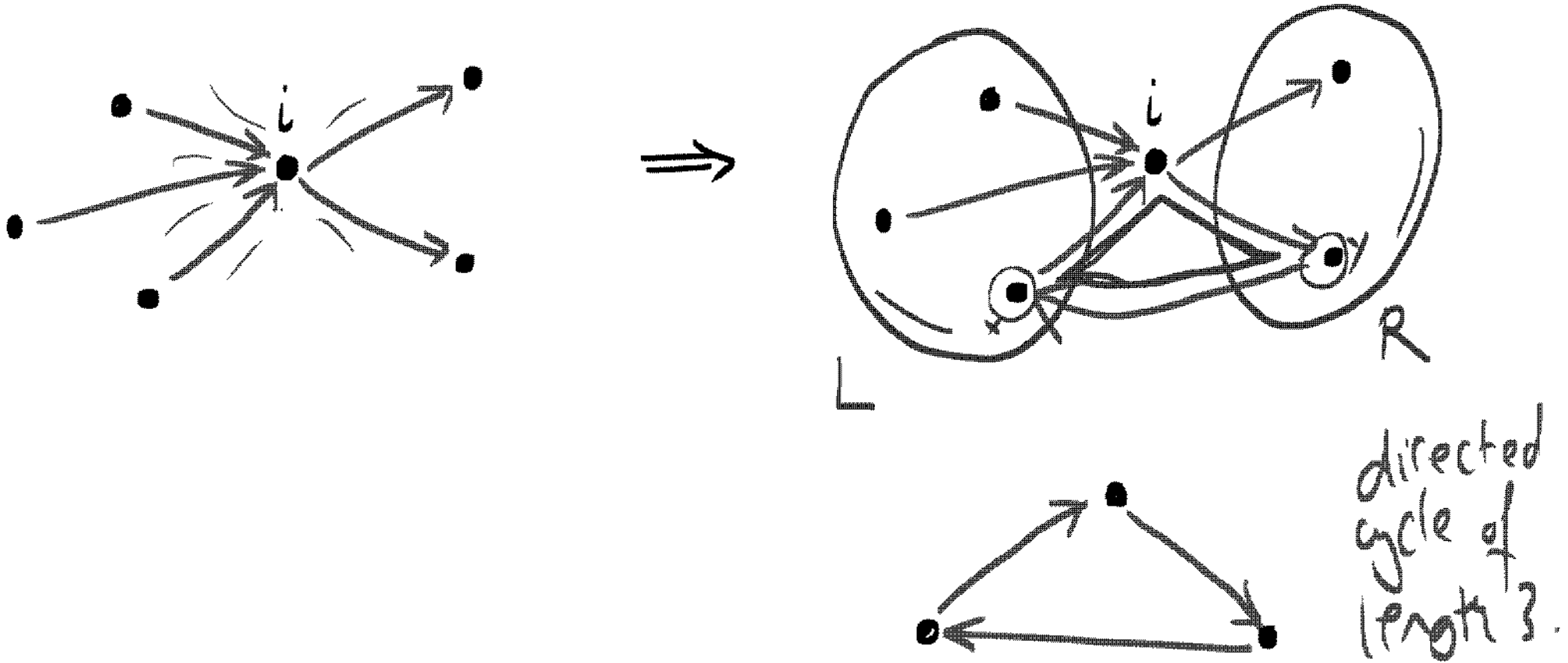
and $R = \{\text{vertices } j \text{ such that } (i,j) \text{ is in } E\}$

Recursively solve the problem for L and for R

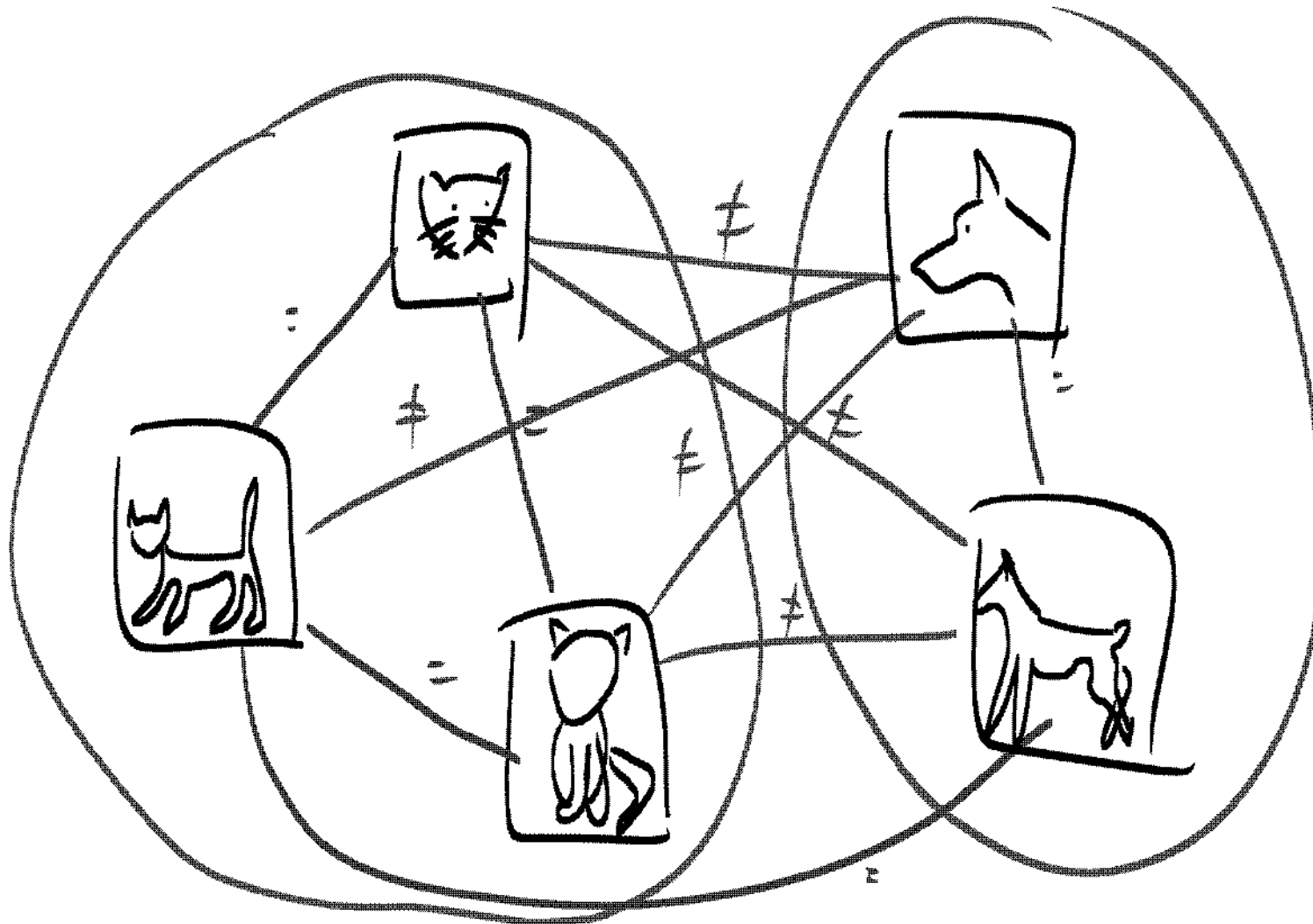
Output: (ordering of L , i , ordering of R)

Theorem [Ailon Charikar Newman 2005]: It's a 3-approximation

Proof: intuition



Correlation clustering



Input

Undirected graph $G=(V,E)$ with
pairs of edge weights

$$\forall \{i,j\} \in E \quad w_{ij}^+ \geq 0 \quad w_{ij}^- \geq 0$$

measures similarity
between i and j

measures dissimilarity
between i and j

Output

partition V "according to similarity"

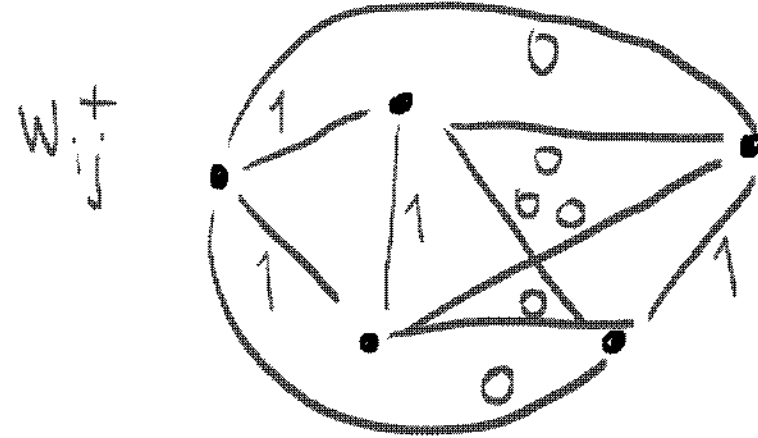
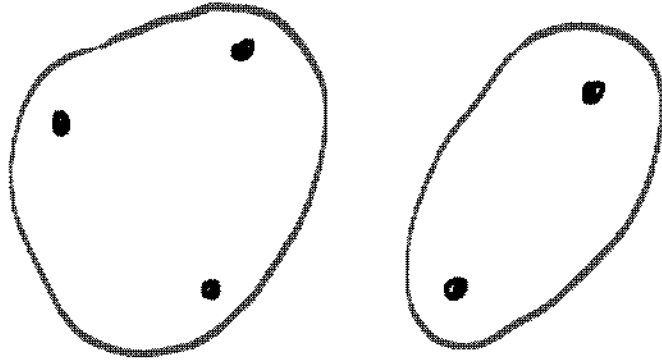
Objective

$$\max \sum_{\{i,j\} \in E} \left[w_{ij}^+ \mathbb{1} \left(\begin{array}{c} i,j \text{ in} \\ \text{same} \\ \text{part} \end{array} \right) + w_{ij}^- \mathbb{1} \left(\begin{array}{c} i,j \text{ in} \\ \text{different} \\ \text{parts} \end{array} \right) \right]$$

Easy cases

If input is perfect...

"easy" input: complete graph, $(w_{ij}^+, w_{ij}^-) = (1, 0)$ or $(0, 1)$
that corresponds exactly to a clustering



Greedy algorithm

Repeat

Pick a vertex i

Form a cluster with all vertices j s.t. $w_{ij}^+ = 1$

equivalent objective: $\min \sum_{\{i,j\} \in E} \left[w_{ij}^+ \mathbb{1} \left(\begin{array}{c} i,j \text{ in} \\ \text{different} \\ \text{parts} \end{array} \right) + w_{ij}^- \mathbb{1} \left(\begin{array}{c} i,j \text{ in} \\ \text{same} \\ \text{part} \end{array} \right) \right]$

Perfect input: value = 0, solvable by greedy algorithm

Near-perfect input: if objective is small

Randomized greedy algorithm

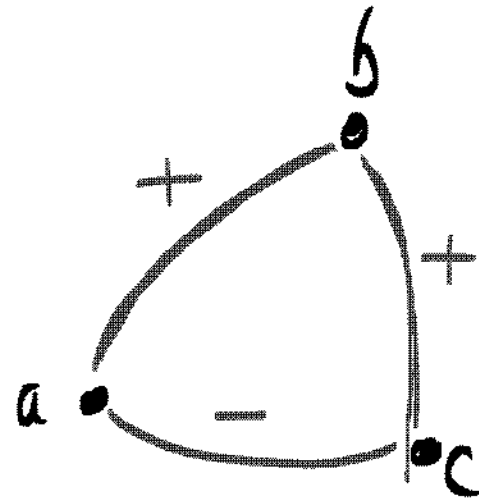
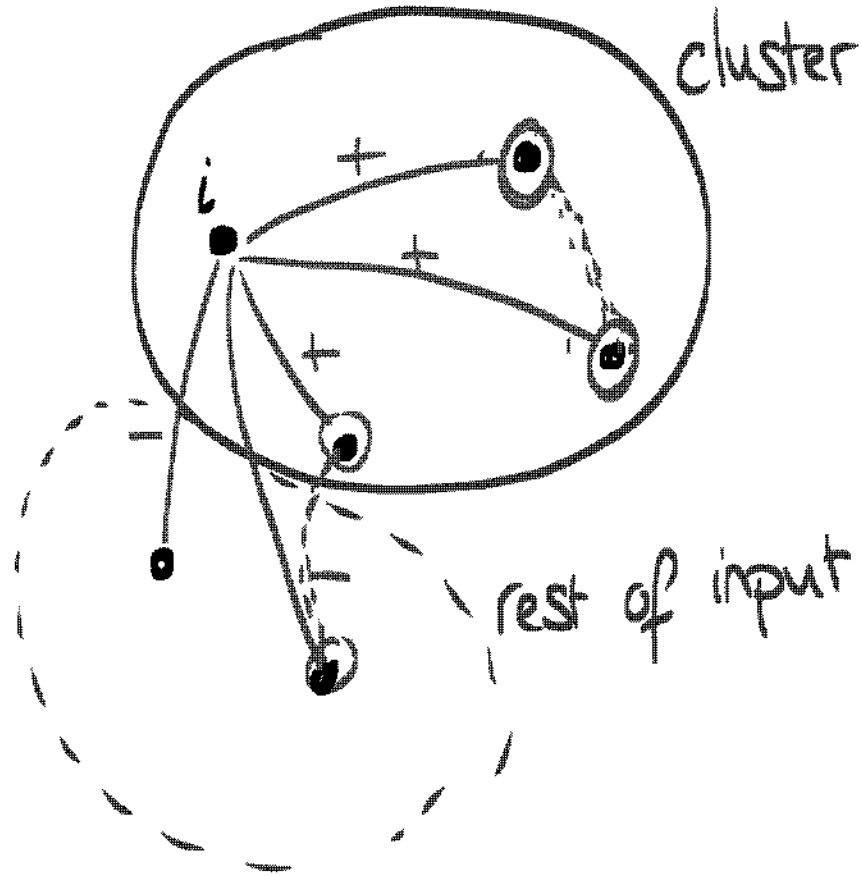
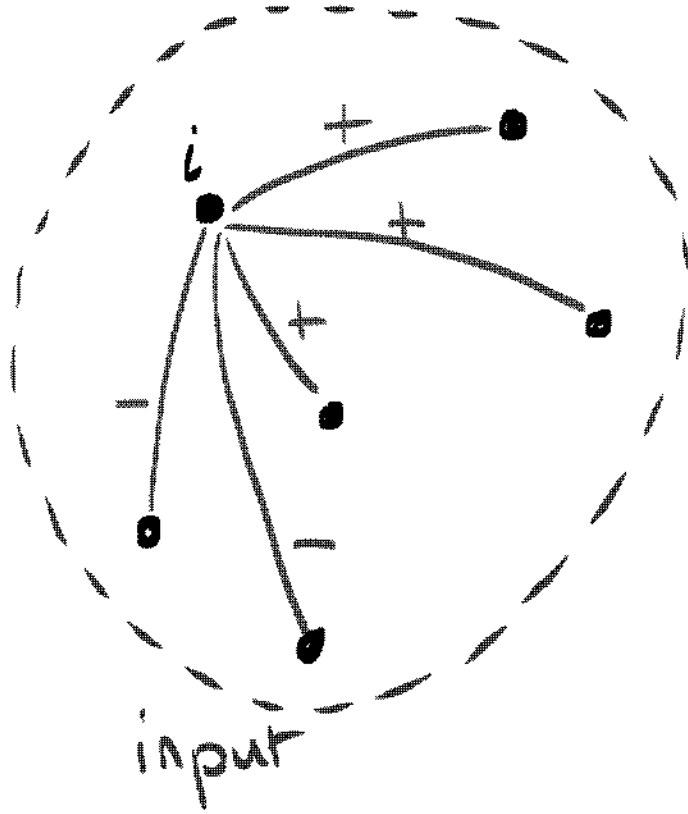
Repeat

Pick a random vertex i

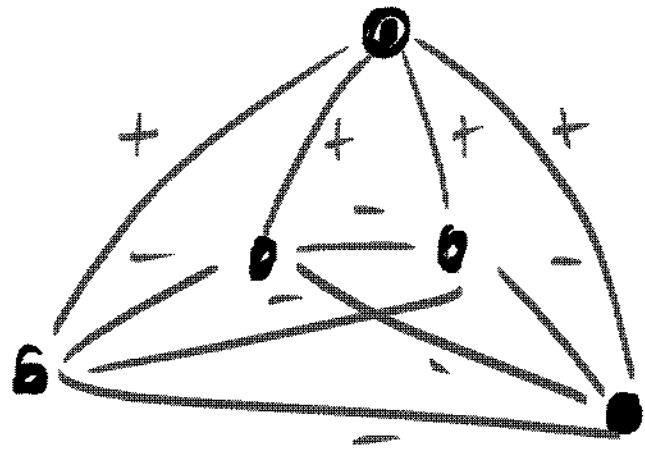
Form a cluster with all vertices j s.t. $w_{ij}^+ = 1$

Theorem [Ailon Charikar Newman 2005]: It's a 3-approximation

Intuition



Why 3?



Conclusion

importance de résolution effective

outils mathématiques
algèbre matricielle
probabilités
(géométrie)

structure