# Chapter 1: Polynomial approximation examples

## MPRO - Complexity: approximation algorithms

Dimitri Watel (dimitri.watel@ensiie.fr)

2023

# The Steiner tree problem

## Inputs

- An undirected graph $G = (V, E)$
- weights $\omega : E \to \mathbb{R}^+$;
- A subset $X \subset V$ of *terminals*

## Output

A tree $T$ included in $G$ and covering $X$ at minimum cost:
min $\bigcup_{e \in T} \omega(e)$.

# Some ideas

Let $k = |X|$ and $n = |V|$

- What do we search if $k = 2$?
- What do we search if $k = n$?

# A 2-approximation algorithm

**Algorithm of Choukkmane (1978) et de Kou *et al.* (1981), PLesnìk (1981) and Iwainsky *et al.* (1986)**

- For every $x, x'$, compute the shortest path $p(x, x')$ from $x$ to $x'$ of weight $d(x, x')$
- Build the complete graph $H = (X, E')$ weighted with $d$.
- Compute a minimum spanning tree $T_H$ of $H$.
- Compute $T = \bigcup_{(x,x') \in T_H} p(x, x')$.
- Simplify $T$ (remove the cycles, the non terminal leaves, ...).
- Return $T$.

Approximation ratio: $2 \cdot (1 - \frac{1}{k})$

# The traveling salesman problem

## Inputs

- A complete undirected graph $G = (V, E)$
- weights $\omega : E \to \mathbb{R}^+$.

## Output

A hamiltonian cycle $C$ of $G$ at minimum cost: $\min \bigcup_{e \in C} \omega(e)$.

# Some ideas

Let $k = |X|$ and $n = |V|$

- What do we search if we remove an edge from an optimal solution?

# A 2-approximation algorithm

**(Simplified) algorithm of Christofides**

- Compute a minimum spanning tree $T$ of $G$.
- Return the cycle that visits the nodes of $G$ in the same order than a depth first search of $T$.

Approximation ratio: $\infty$
Approximation ratio: 2 si inégalité triangulaire.

# A $\frac{3}{2}$-approximation algorithm

## Algorithm of Christofides

- Compute a minimum spanning tree $T$ of $G$.
- Let $V'$ be the odd degree nodes of $T$, compute (in $G$) a minimum cost perfect matching $M$ of $V'$.
- Compute an eulerian cycle $C'$ of $T \cup M$ and return the cycle that visits the nodes of $G$ in the same order than $C'$.

Approximation ratio: $\infty$
Approximation ratio: $\frac{3}{2}$ si inégalité triangulaire.

# The set covering problem

## Inputs

- A set $X$;
- a set of parts of $X$: $S \subset 2^X$;
- weights $\omega : S \to \mathbb{R}^+$;

## Output

A subset $C \subset S$ covering $X$ (*i.e.* $X \subset \bigcup_{s \in C} s$) at minimum cost:

$$\min \sum_{s \in C} \omega(s).$$

# Version non pondérée

**Inputs**

- A set $X$;
- a set of parts of $X$: $S \in 2^X$;

**Output**

A subset $C \subset S$ covering $X$ with minimum size: $\min |C|$.

Which sets would you intuitively choose?

# A $\ln(k)$-approximation algorithm (unweighted version)

**Algorithm of Johnson (1974), Lovàsz (1975) and Chvàtal (1979)**

- While some elements of $X$ are uncovered, select the set containing the maximum number of uncovered elements.

Approximation ratio: $\ln(k/OPT) + 1$

# A $\ln(k)$-approximation algorithm (weighted version)

**Algorithm of Johnson (1974), Lovàsz (1975) and Chvàtal (1979)**

- While some elements $X'$ of $X$ are uncovered, select the set $s$ minimizing $\omega(s)/|X' \cap s|$.

Approximation ratio: $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k} \sim \ln(k)$

# The knapsack problem

## Inputs

- An integer $V$ (the volume of the bag);
- entiers $u_1, u_2, \dots, u_n$ (the utilities of the objects);
- entiers $v_1, v_2, \dots, v_n$ (the volumes of the objects).

## Output

Objects that can be put in the bag and maximizing the value: A subset $I \subset [\![1; n]\!]$ such that $\sum_{i \in I} v_i \leq V$ and maximizing $\sum_{i \in I} u_i$.

# Some ideas

The knapsack problem is pseudo-polynomial.

- if the volume of the bag is polynomial, the problem is polynomial
- if the volume of the objects are polynomial, the problem is polynomial
- if the profits of the objects are polynomial, the problem is polynomial

# An FPTAS

## Algorithm of Ibarra and Kim (1975)

- Let $\varepsilon > 0$ and $K = \frac{\varepsilon \max(u_i)}{n}$
- Let $u'_i = \lfloor \frac{u_i}{K} \rfloor$
- Solve the instance where each utility $u_i$ is replaced by $u'_i$ and return the solution found.

Approximation ratio: $1 - \varepsilon$
Time complexity: $O(n^2 \lfloor \frac{n}{\varepsilon} \rfloor)$

# The bin packing problem

## Inputs

- An integer $V$ (the volume of a bin);
- entiers $v_1, v_2, \ldots, v_n$ (the volumes of the objects).

## Output

The minimum number of bins that are needed to store every object : $n$ integers $a_1, a_2, \ldots, a_n$ such that $\forall i, \sum_{j : a_j = i} v_j \leq V$ minimizing $\max(a_i)$.

Remarque : on peut supposer $V = 1$

## Some ideas

- What if I receive the objects in an arbitrary order and if I have to immediately decide where to put it?

# A 2-approximation

**Algorithme First fit**

For every $i \in [\![1; n]\!]$, put $i$ in the first valid bin.

Approximation ratio: 2

# Some other ideas

- If there exists two constants $\varepsilon$ and $K$ such that every object has size at least $\varepsilon$ and if the number of distinct objects is $K$, the problem is polynomial.

**Lemme**

If every object has size at least $\varepsilon$, there exists an $(1+\varepsilon)$ approximation algorithm:

- Sort every objects
- Group them, in that order, in $\lceil \frac{1}{\varepsilon^2} \rceil$ groups of size at most $\lfloor n\varepsilon^2 \rfloor$.
- Replace every object $x$ by $x'$, the largest item in its group.
- Find an optimal solution $S$ of that new instance.
- Return the same solution in the first instance (after replacing

# An asymptotic PTAS

## Vega and Lueker algorithm

- Let $\varepsilon > 0$
- Let $A$ be all the objects of size lower than $\varepsilon$. Remove $A$.
- Find a $1 + \varepsilon$ approximated solution with the algorithm of the previous lemma.
- Add every object of $A$ with the First Fit algorithm.

Approximation ratio: $(1 + 2\varepsilon + \frac{1}{OPT})$

Complexity: $O(n^{\binom{M+K}{M}})$, $M = \lfloor \frac{1}{\varepsilon} \rfloor$, $K = \lceil \frac{1}{\varepsilon^2} \rceil$

# Minimum Planar Coloring problem

## Inputs

- A graph $G = (V, E)$ planaire;

## Output

The minimum number of colors needed to color $G$

# Quelques idées

- A graph can be colored with 1 color if and only if it is disconnected
- A graph can be colored with 1 color if and only if it is bipartite
- A planar graph can be colored with 4 colors [Appel, Haken (1976)]

# Une approximation de rapport absolu 1

## Algorithme de coloration de graphe

- Si le graphe est déconnecté, renvoyer 1
- Si le graphe est biparti, renvoyer 2
- Sinon renvoyer 4

Rapport d'approximation (absolu) : 1