Chapter 2: Optimization problems and approximation algorithms

MPRO - Complexity: approximation algorithms

Dimitri Watel (dimitri.watel@ensiie.fr)

2024

Decision problem

(Informal) definition

A decision problem is a problem describing inputs (*the instances*) and a question about the input for which the answer is **Yes** or **No** and depends on the input.

The complexity class P

Definition of the class P

A decision problem Π is polynomial, or belongs to P, if its complexity is polynomial. In other words, there is a constant c such that its complexity is $O(n^c)$.

The complexity class EXPTIME

Definition of the class EXPTIME

A decision problem Π is exponential, or belongs to EXPTIME, if its complexity is exponential. In other words, there is a constant c such that its complexity is $O(2^{(n^c)})$.

$$\text{P} \subsetneq \text{EXPTIME}$$

The complexity class NP

Definition of the class NP

A decision problem Π is belongs to NP if there exists a constant c and a verifier $\mathcal{V}$ for the answers **YES** with complexity $O(n^c)$ and $O(n^c)$.

!!!

$$P \subset NP \subsetneq EXPTIME$$

The complexity class P

Definition of the class P

A decision problem Π is polynomial, or belongs to P, if its complexity is polynomial. In other words, there is a **deterministic** Turing machine that solves Π in polynomial time.

$$\mathsf{P} = \bigcup_{c \in \mathbb{N}} \mathsf{DTIME}(n^c)$$

The complexity class EXPTIME

Definition of the class EXPTIME

A decision problem Π is exponential, or belongs to EXPTIME, if its complexity is exponential. In other words, there is a **deterministic** Turing machine that solves Π in exponential time:

$$\text{EXPTIME} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{n^c})$$

$$\text{P} \subsetneq \text{EXPTIME}$$

The complexity class NP

Definition of the class NP (non-deterministic Polynomial)

A decision problem Π belongs to NP, if there is a **non-deterministic** Turing machine that solves Π in polynomial time.

$$\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)$$

$$\text{P} \subset \text{NP} \subset \text{PSPACE}$$

Karp polynomial reduction

(Informal) definition

Let Π_1 and Π_2, a polynomial reduction from Π_1 to Π_2 transforms every positive (resp. negative) instance of Π_1 into a positive (resp. negative) instance of Π_2 in polynomial time.

If Π_1 reduces to Π_2, then we say that Π_2 is *harder* than Π_1 and we write:

$$\Pi_1 \preccurlyeq \Pi_2$$

Hardness

NP-Hard problem

Let Π_1 be a decision problem. Π_1 is NP-Hard if, for every problem Π_2 of $\mathcal{C}$, $\Pi_2 \preccurlyeq \Pi_1$.

Completeness

NP-complete problem

Let Π be a decision problem. Π is NP-Complete if $\Pi \in NP$ and if it is NP-Hard.

Turing polynomial reduction

Definition: Turing polynomial reduction

Let Π_1 and Π_2 bet two problems, a Turing polynomial reduction from Π_1 to Π_2 is an algorithme $\mathcal{R}$ such that

- $\mathcal{R}$ may call an (imaginary) algorithm $\mathcal{R}'$ called *oracle* that solves Π_2 in constant time;
- $\mathcal{R}$ solves Π_1 in polynomial time.

We write $\Pi_1 \preccurlyeq_T \Pi_2$.

Π_1 and Π_2 are not necessarily decision problems.

Optimization problem

(Informal) definition

An optimization problem is a problem containing inputs (the *instances*), solutions (the *feasible solutions*), a *measure* or *weight* associating to each solution an integer. The objective is to find a solution maximizing or minimizing the measure.

Problems associated with an optimization problem

(Informal) definition

An optimization problem Π is associated with 3 problems:

- a *decision problem* Π_D: does there exist a feasible solution with weight
  - lower than K?, if the objective is to minimize the measure,
  - greater than K?, if the objective is to maximize the measure ;
- an *evaluation problem* Pi_E: find the weight of an optimal solution;
- a *construction problem* Π_C: find an optimal solution and its weight.

PO and NPO

Definition

PO and NPO are the equivalent classes of P and NP for the optimization problems: the optimization problems that can be solved in deterministic or non deterministic polynomial time.

Particularly,

- $\Pi \in PO \Rightarrow \Pi_D \in P$
- $\Pi \in NPO \Rightarrow \Pi_D \in NP$

Optimization problem

Definition: optimization problem

An optimization problem Π is a quadruplet $(\mathcal{I}, \mathcal{S}, \mathcal{M}, \mathcal{O})$ such that:

- $\mathcal{I}$ is the set of d'*instances* of Π ;
- $\mathcal{S}$ is a function associating to $x \in \mathcal{I}$ a set of *feasible solutions* of x ;
- $\mathcal{M}$ is a *measure* function associating to $x \in \mathcal{I}$ and $y \in \mathcal{S}(x)$ an integer ;
- $\mathcal{O}$ is the *objectif* with value min or max that specify whether we want to find a solution with minimum of maximum measure.

$\mathcal{I}$ and $\mathcal{S}(x)$ may contain any mathematical objects.

Optimal solution

Definition

Let $\Pi = (\mathcal{I}, \mathcal{S}, \mathcal{M}, \mathcal{O})$ be an optimization problem and $x \in \mathcal{I}$. We call *optimal solution* a solution $y^* \in \mathcal{S}(x)$ such that:

$$\mathcal{M}(x, y^*) = \min_{y \in \mathcal{S}(x)} \mathcal{M}(x, y) \text{ si } \mathcal{O} = \min$$

$$\mathcal{M}(x, y^*) = \max_{y \in \mathcal{S}(x)} \mathcal{M}(x, y) \text{ si } \mathcal{O} = \max$$

We denote the *set of optimal solutions* of x by $\mathcal{S}^*(x)$ and the value $\mathcal{M}(x, y^*)$ of the optimal solutions by $\mathcal{M}^*(x)$.

Size of an instance or of a solution

Definition

The size of an instance or of a feasible solution is the number of bits used to encode it.

We usually denote the size of the input x by $|x|$ or n and the size of a feasible solution y by $|y|$.

Problems associated with an optimization problem

Definition

An optimization problem $\Pi = (\mathcal{I}, \mathcal{S}, \mathcal{M}, \mathcal{O})$ is associated with 3 problems:

- a *decision problem* Π_D: let $x \in \mathcal{I}$ and an integer K, determine if $\mathcal{M}^*(x) \leq K$ if $\mathcal{O} = \min$, or if $\mathcal{M}^*(x) \geq K$ if $\mathcal{O} = \max$.
- an *evaluation problem* Pi_E: let $x \in \mathcal{I}$, compute $\mathcal{M}^*(x)$;
- a *construction problem* Π_C: let $x \in \mathcal{I}$, compute an optimal solution $y \in \mathcal{S}^*(x)$ and $\mathcal{M}^*(x)$.

Some results

Theorem

Let Π be an optimization problem:

$$\Pi_D \preccurlyeq_T \Pi_E \preccurlyeq_T \Pi_C$$

$$\Pi_E \preccurlyeq_T \Pi_D \text{ si } \mathcal{M}^*(x) = O(2^{|x|^c})$$

NPO

Definition

Let $\Pi = (\mathcal{I}, \mathcal{S}, \mathcal{M}, \mathcal{O})$ be an optimization problem, Π belong to the NPO class if

- let x be a binary number, we can check in polynomial time if x encodes an instance of $\mathcal{I}$;
- there exists a polynom q such that for every $x \in \mathcal{I}$
  - for every $y \in \mathcal{S}(x)$, $|y| \leq q(|x|)$,
  - for every binary number y such that $|y| \leq q(|x|)$, we can check in polynomial time if y encodes a feasible solution of x;
- for every $x \in \mathcal{I}$ and every $y \in \mathcal{S}(x)$, we can compute $\mathcal{M}(x, y)$ in polynomial time.

PO

Definition

A problem $\Pi \in$ NPO belongs to the class PO if we can solve Π_C in polynomial time.

By definition PO $\subseteq$ NPO.

PO, NPO, P and NP

Theorem

$$\Pi \in PO \Rightarrow \Pi_D \in P$$

$$\Pi \in NPO \Rightarrow \Pi_D \in NP$$

NP-Hard optimization problem

Definition

Let Π_1 be an optimization problem, Π_1 is NP-Hard, for every decision problem $\Pi_2 \in NP$, $\Pi_2 \preccurlyeq_T \Pi_1$.

Theorem

Let $\Pi \in NPO$, if Π_D is NP-Hard, then Π is NP-Hard.

Theorem

$PO = NPO \Rightarrow P = NP$

Polynomial approximation

Do we have to return an optimal solution?

Polynomial approximation

Definition

A *polynomial approximation algorithm* for a problem Π is a polynomial algorithm that returns a feasible solution of Π that is "close" to an optimal solution.

Polynomial approximation of a minimization problem

Definition

A *polynomial approximation algorithm* for a minimization problem Π with ratio r or r polynomial approximation algorithm is a polynomial algorithm that, for every instance x of Π, returns a feasible solution of Π such that $M(x, y) \leq r(x) \cdot M^*(x)$.

r est une fonction de $\mathcal{I}$ dans $\mathbb{R}$.

Polynomial approximation of a maximization problem

Definition

A *polynomial approximation algorithm* for a maximization problem Π with ratio r or r polynomial approximation algorithm is a polynomial algorithm that, for every instance x of Π, returns a feasible solution of Π such that $M(x, y) \geq r(x) \cdot M^*(x)$.

r est une fonction de $\mathcal{I}$ dans $\mathbb{R}$.

Approximability classes: APX

Definition

A problem $\Pi \in$ NPO belongs to the class APX if there exists a constant c and a c-polynomial approximation for Π.

$$\text{APX} \subset \text{NPO (Si P} \neq \text{NP, APX} \subsetneq \text{NPO.)}$$

Approximability classes: r-APX

Soit r une fonction ($r : \mathbb{N} \rightarrow \mathbb{R}^+$)

Definition

A problem $\Pi \in$ NPO belongs to the class r-APX if there exists a $r(|\mathcal{I}|)$-polynomial approximation for Π.

De maninère similaire, on définit la classe $O(r)$-APX.
r-APX $\subset$ NPO (Si P $\neq$ NP, r-APX $\subsetneq$ NPO.)

Approximability classes: PTAS

Definition : PTAS

A problem $\Pi \in$ NPO belongs to the class FPTAAS if there exists an approximation scheme for Π: for every $\varepsilon > 0$, there exists a $(1+\varepsilon)$-polynomial approximation for Π.

PTAS $\subset$ APX (Si P $\neq$ NP, PTAS $\subsetneq$ APX.)

Approximability classes: FPTAS

Definition : FPTAS

A problem $\Pi \in$ NPO belongs to the class FPTAS if there exists a fully polynomial time approximation scheme for Π: for every $\varepsilon > 0$, there exists a $(1+\varepsilon)$-polynomial approximation for Π with a complexity polynomial in the size of the input and in $\frac{1}{\varepsilon}$.

FPTAS $\subset$ PTAS (Si P $\neq$ NP, FPTAS $\subsetneq$ PTAS.)

Approximability classes: EPTAS

Definition : EPTAS

A problem $\Pi \in$ NPO belongs to the class EPTAS if there exists an efficient approximation scheme for Π: for every $\varepsilon > 0$, there exists a $(1 + \varepsilon)$-polynomial approximation for Π with a complexity $O(f(\frac{1}{\varepsilon}) \cdot n^c)$.

FPTAS $\subset$ EPTAS $\subset$ PTAS (Si P $\neq$ NP, FPTAS $\subsetneq$ EPTAS, et si ??? $\neq$??? (cf cours de complexité paramétrée), EPTAS $\subsetneq$ PTAS.)

Peut-on caractériser les problèmes qui ont ou n'ont pas de PTAS/FPTAS?

Binary encoding VS Unary encoding

Binary encoding

Encode an integer k with *binary encoding* on a Turing machine consists in writing using the base-2 system with $\log_2(k)$ bits surrounded by two white symbols. The memory space used by k is $log_2(k) + 2$.

Unary encoding

Encode an integer k with *unary encoding* on a Turing machine consists in writing with k cells filled with a 1 surrounded by two white symbols. The memory space used by k is $k + 2$.

Complexity and encoding

Observation

Let Π be a decision problem, x be an instance of Π containing an integer W and $\mathcal{A}$ be an algorithm with complexity $O(W)$ solving Π.

- If W is unary encoded, $\mathcal{A}$ is polynomial.
- If W is binary encoded, $\mathcal{A}$ is exponential.

Strong NP-Completeness

Definition

Let Π be a decision problem, we say that Π is *strongly NP-Complete* if it is NP-Complete when we encode the integers of the instance of Π with unary encoding.

Weak NP-Completeness

Definition

Let Π be a decision problem, we say that Π is *weakly NP-Complete* if it is NP-Complete and if there exists a polynomial algorithm to solve it when we encode the integers of the instance of Π with unary encoding.

Usual size of the input: 2*nd* try.

Definition

Let x be an input containing a set x_0 of non numerical objects and a set of k integers $(x_1, x_2, \dots, x_k)$, we define the size of x in multiple ways :

- the total usual size $|x|$: the number of bits we need to encode it;
- the size $l(x)$ that does not depend on the values of the integers : $|x_0| + k$;
- the size $\max(x)$ related to the integers : $\max\limits_{[\![1;k]\!]} x_i$.

Strong NP-Completeness: alternative definition

Definition

Let Π be a decision problem, we say that Π is *strongly NP-Complete* if it is NP-Complete when the instances are restricted to the ones where $\max(x)$ is polynomially bounded by $l(x)$.

Theorem

The two definitions of strong NP-Completeness are equivalent.

Weak NP-Completeness

Definition: pseudo-polynomial complexity

Let Π be a decision problem, an algorithm solving Π is *pseudo-polynomial* if its time complexity is polynomial in $l(x)$ and $\max(x)$.

Definition

Let Π be a decision problem, we say that Π is *weakly NP-Complete* if it is NP-Complete and if there exists a pseudo-polynomial algorithm to solve it.

Theorem

The two definitions of weak NP-Completeness are equivalent.

Strong $\neq$ Weak

Theorem

If P $\neq$ NP then a strongly NP-Complete problem is not weakly NP-Complete and conversely.

Show that a problem is strongly NP-Complete.

Theorem

Let Π be an NP-Complete problem containing no integer, then Π is strongly NP-Complete.

Theorem

Let Π_1 be a strongly NP-Complete problem and Π_2 be a decision problem. If there exists a polynomial Karp reduction such that, for every instance x of Π_1, x is transformed into an instance y of Π_2 such that $\max(y)$ is polynomially bounded in $l(x)$ and/or $\max(x)$, then Π_2 is strongly NP-Complete.

Strong NPO

Definition

Let $\Pi \in \mathrm{NPO}$ be an optimization problem, we say that Π is *strongly NP-Hard* if it is NP-Hard when the instances are restricted to the ones where $\max(x)$ is polynomially bounded by $I(x)$.

Weak NPO

Definition

Let $\Pi \in$ NPO be an optimization problem, we say that Π is *weakly NP-Hard* if it is NP-Hard and if there exists a pseudo-polynomial algorithm to solve it.

FPTAS and pseudo-polynomial algorithm

Theorem

Let $\Pi \in$ FPTAS such that $\mathcal{M}^*(x)$ is polynomially bounded by $l(x)$ and $\max(x)$, then there exists a pseudopolynomial algorithm that solves Π.

Corollary

Let $\Pi \in$ NPO be a strongly NP-Hard problem such that $\mathcal{M}^*(x)$ is polynomially bounded by $l(x)$, then $\Pi \notin$ FPTAS.

Rapport d'approximation absolu

Definition

An **absolute** *polynomial approximation algorithm* for a minimization problem Π with absolute ratio r is a polynomial algorithm that, for every instance x of Π, returns a feasible solution of Π such that $M(x, y) \leq M^*(x) + r(x)$.

r est une fonction de $\mathcal{I}$ dans $\mathbb{R}$.

Rapport d'approximation absolu

Definition

An **absolute** *polynomial approximation algorithm* for a maximization problem Π with absolute ratio r is a polynomial algorithm that, for every instance x of Π, returns a feasible solution of Π such that $M(x,y) \geq M^*(x) - r(x)$.

r est une fonction de $\mathcal{I}$ dans $\mathbb{R}$.

Approximability classes: AAPX

Definition

A problem $\Pi \in$ NPO belongs to the class AAPX if there exists a constant c and a c-absolute polynomial approximation for Π.

AAPX $\subset$ NPO (Si P $\neq$ NP, AAPX $\not\subset$ PTAS et PTAS $\not\subset$ AAPX.)
On définit de même APTAS, AFPTAS, A-$O(r)$-APX, ...

Rapport d'approximation asymptotique

Definition

An **asymptotical** *polynomial approximation algorithm* for a minimization problem Π with absolute ratio r is a polynomial algorithm that, for every instance x of Π, returns a feasible solution of Π such that $M(x, y) \leq M^*(x) \cdot r(x) + k$.

r est une fonction de $\mathcal{I}$ dans $\mathbb{R}$.

Rapport d'approximation asymptotique

Definition

An **asymptotical** *polynomial approximation algorithm* for a maximization problem Π with absolute ratio r is a polynomial algorithm that, for every instance x of Π, returns a feasible solution of Π such that $M(x, y) \geq M^*(x, y) \cdot r(x) - k$.

r est une fonction de $\mathcal{I}$ dans $\mathbb{R}$.

Approximability classes: APX_∞

Definition

A problem $\Pi \in$ NPO belongs to the class APX_∞ if there exists a constant c and a c-asymptotical polynomial approximation for Π.

$$\mathrm{APX}_\infty = \mathrm{APX}$$

On définit de même PTAS_∞, FPTAS_∞, $O(r)$-APX_∞, ...

Si P $\neq$ NP, $\mathrm{PTAS}_\infty \neq \mathrm{PTAS}$.

Autres rapports

- Rapport moyen : se comparer à la moyenne des valeurs des solutions réalisables
- Différentiel : biaiser le rapport solution/optimal avec le poids ω d'une pire solution : $\frac{M(x,y)-\omega}{M^*(x)-\omega}$.