Approximation and paremeterization: Complexity classes and hard problems

Maria Serna

Spring 2025

FPT-reductions

FPT-reductions

- Let (L, κ) and (L', κ') be two parameterized problems (on the same alphabet Σ)

FPT-reductions

- Let (L, κ) and (L', κ') be two parameterized problems (on the same alphabet Σ)
- A FPT-reduction from (L, κ) to (L', κ') is a mapping $R : \Sigma^* \to \Sigma^*$ where

FPT-reductions

- Let (L, κ) and (L', κ') be two parameterized problems (on the same alphabet Σ)
- A FPT-reduction from (L, κ) to (L', κ') is a mapping $R : \Sigma^* \to \Sigma^*$ where
  - $\forall x \in \Sigma^* \; x \in L$ iff $R(x) \in L'$
  - There is an FPT-algorithm with respect to κ computing R (in $f(\kappa(x))p(|x|)$)
  - There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $\forall x \in \Sigma^* \kappa'(R(x)) \leq g(\kappa(x))$

FPT-reductions

- Let (L, κ) and (L', κ') be two parameterized problems (on the same alphabet Σ)
- A FPT-reduction from (L, κ) to (L', κ') is a mapping $R : \Sigma^* \to \Sigma^*$ where
  - $\forall x \in \Sigma^* \; x \in L$ iff $R(x) \in L'$
  - There is an FPT-algorithm with respect to κ computing R (in $f(\kappa(x))p(|x|)$)
  - There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $\forall x \in \Sigma^* \kappa'(R(x)) \leq g(\kappa(x))$
- We note $(L, \kappa) \leq^{fpt} (L', \kappa')$ when there is a FPT-reduction from (L, κ) to (L', κ')

FPT-reductions

Lemma

FPT is closed under FPT-reductions

FPT-reductions

Lemma

FPT is closed under FPT-reductions

- We have to show that
  If $(L', \kappa') \in$ FPT and $(L, \kappa) \leq^{fpt} (L', \kappa')$ then $(L, \kappa) \in$ FPT

FPT-reductions

Lemma

FPT is closed under FPT-reductions

- We have to show that
  If $(L', \kappa') \in$ FPT and $(L, \kappa) \leq^{fpt} (L', \kappa')$ then $(L, \kappa) \in$ FPT
- Algorithm $\mathcal{A}'$ solves (L', κ') in $f'(\kappa'(y))p(|y|)$ time.
- Computing the FPT reduction R takes time $f_R(\kappa(x))p_R(|x|)$.
- Running $\mathcal{A}'$ on $y = R(x)$ solves (L, κ) in time

$$\begin{aligned} f_R(\kappa(x))p_R(|x|) + f'(\kappa'(R(x)))p(|R(x)|) \\ &\leq f_R(\kappa(x))p_R(|x|) + f'(g(\kappa(x))p(|R(x)|) \\ &\leq f_R(\kappa(x))p_R(|x|) + f'(g(\kappa(x))p(f_R(\kappa(x))p_R(|x|)) \\ &\leq f(\kappa(x))p(|x|) \end{aligned}$$

FPT-reductions and complexity classes

FPT-reductions and complexity classes

- FPT-equivalence
  $(L, \kappa) \equiv^{fpt} (L', \kappa')$: $(L, \kappa) \leq^{fpt} (L', \kappa')$ and $(L', \kappa') \leq^{fpt} (L, \kappa)$

FPT-reductions and complexity classes

- FPT-equivalence
  $(L, \kappa) \equiv^{fpt} (L', \kappa')$: $(L, \kappa) \leq^{fpt} (L', \kappa')$ and $(L', \kappa') \leq^{fpt} (L, \kappa)$
- Closure under FPT-reductions
  $[(L, \kappa)]^{fpt} = \{(L', \kappa') \mid (L', \kappa') \leq^{fpt} (L, \kappa)\}$

FPT-reductions and complexity classes

- FPT-equivalence
  $(L, \kappa) \equiv^{fpt} (L', \kappa')$: $(L, \kappa) \leq^{fpt} (L', \kappa')$ and $(L', \kappa') \leq^{fpt} (L, \kappa)$
- Closure under FPT-reductions
  $[(L, \kappa)]^{fpt} = \{(L', \kappa') \mid (L', \kappa') \leq^{fpt} (L, \kappa)\}$
- If $\mathcal{C}$ is a class of parameterized problems
  - (L, κ) is $\mathcal{C}$-hard if $\mathcal{C} \subseteq [(L, \kappa)]^{fpt}$.
  - (L, κ) is $\mathcal{C}$-complete if $(L, \kappa) \in \mathcal{C}$ and (L, κ) is $\mathcal{C}$-hard.

FPT-reductions and complexity classes

- FPT-equivalence
  $(L, \kappa) \equiv^{fpt} (L', \kappa')$: $(L, \kappa) \leq^{fpt} (L', \kappa')$ and $(L', \kappa') \leq^{fpt} (L, \kappa)$
- Closure under FPT-reductions
  $[(L, \kappa)]^{fpt} = \{(L', \kappa') \mid (L', \kappa') \leq^{fpt} (L, \kappa)\}$
- If $\mathcal{C}$ is a class of parameterized problems
  - (L, κ) is $\mathcal{C}$-hard if $\mathcal{C} \subseteq [(L, \kappa)]^{fpt}$.
  - (L, κ) is $\mathcal{C}$-complete if $(L, \kappa) \in \mathcal{C}$ and (L, κ) is $\mathcal{C}$-hard.
- $[(L, \kappa)]^{fpt}$ defines a class of parameterized problems for which (L, κ) is complete
- if (L, κ) is $\mathcal{C}$-complete and $\mathcal{C}$ is closed under FPT reductions, then $\mathcal{C} = [(L, \kappa)]^{fpt}$

FPT-equivalent problems

- p-INDEPENDENT SET $\equiv^{fpt}$ p-CLIQUE

FPT-equivalent problems

- p-Independent Set $\equiv^{fpt}$ p-Clique

$$R(G, k) = (\overline{G}, k)$$

  Works for both directions
- p-Hitting Set $\equiv^{fpt}$ p-Dominating Set

FPT-equivalent problems

- p-Independent Set $\equiv^{fpt}$ p-Clique

$$R(G,k) = (\overline{G}, k)$$

  Works for both directions
- p-Hitting Set $\equiv^{fpt}$ p-Dominating Set
  Exercise

The class paraNP

- Let (L, κ) be a parameterized problem
- (L, κ) belongs to paraNP if there is a non-deterministic algorithm $\mathcal{A}$ that decides $x \in L$ in time $f(\kappa(x))p(|x|)$, for some computable function f and polynomial function p.

The class paraNP

- Let (L, κ) be a parameterized problem
- (L, κ) belongs to paraNP if there is a non-deterministic algorithm $\mathcal{A}$ that decides $x \in L$ in time $f(\kappa(x))p(|x|)$, for some computable function f and polynomial function p.
- If $L \in$ NP, for each parameterization κ, $(L, \kappa) \in$ paraNP
  p-Clique, p-Vertex Cover, ... belong to paraNP.

paraNP-completeness

paraNP-completeness

- Let (L, κ) be a parameterized problem

paraNP-completeness

- Let (L, κ) be a parameterized problem
- (L, κ) is trivial if $L = \emptyset$ or $L = \Sigma^*$.

paraNP-completeness

- Let (L, κ) be a parameterized problem
- (L, κ) is trivial if $L = \emptyset$ or $L = \Sigma^*$.
- The *i*-th slice of (L, κ) is the decision problem $(L, \kappa)_i = \{x \in L \mid \kappa(x) = i\}$

paraNP-completeness

- Let (L, κ) be a parameterized problem
- (L, κ) is trivial if $L = \emptyset$ or $L = \Sigma^*$.
- The *i*-th slice of (L, κ) is the decision problem $(L, \kappa)_i = \{x \in L \mid \kappa(x) = i\}$

Theorem

If $(L, \kappa) \in$ paraNP is not trivial and has a NP-complete slice, then (L, κ) is paraNP-complete under FPT reductions.

paraNP-completeness:problems

- p-Vertex Coloring

paraNP-completeness:problems

- p-VERTEX COLORING is paraNP-complete.

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique is not paraNP-complete, unless P $=$ NP.

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique is not paraNP-complete, unless P $=$ NP.
- p#var-Sat

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique is not paraNP-complete, unless P = NP.
- p#var-Sat is not paraNP-complete, unless P = NP.

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique is not paraNP-complete, unless P $=$ NP.
- p#var-Sat is not paraNP-complete, unless P $=$ NP.
- pMax#Lit-Sat

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique is not paraNP-complete, unless P = NP.
- p#var-Sat is not paraNP-complete, unless P = NP.
- pMax#Lit-Sat is paraNP-complete.

paraNP-completeness:problems

- p-Vertex Coloring is paraNP-complete.
- p-Clique is not paraNP-complete, unless P $=$ NP.
- p#var-Sat is not paraNP-complete, unless P $=$ NP.
- pMax#Lit-Sat is paraNP-complete.

- paraNP-completeness separates *all slices* in P from *a slice* is NP-hard.

The class XP

- Let (L, κ) be a parameterized problem.
- (L, κ) belongs to (uniform) XP if there is an algorithm $\mathcal{A}$ that decides L in time $O(|x|^{f(\kappa(x))})$,
  for some computable function f.

The class XP

- Let (L, κ) be a parameterized problem.
- (L, κ) belongs to (uniform) XP if there is an algorithm $\mathcal{A}$ that decides L in time $O(|x|^{f(\kappa(x))})$, for some computable function f.
- p-CLIQUE, p-VERTEX COVER, p-HITTING SET, p-HITTING SET, p-DOMINATING SET belong to XP.
- XP is the counterpart of EXP in classic complexity.

XP-complete problems

p-Exp-DTM-Halt
Input: A deterministic TM $\mathbb{M}$, $x \in \Sigma^*$ and an integer k,
Parameter: k
Question: Does $\mathbb{M}$ on input x stop in no more than $|x|^k$ steps?

XP-complete problems

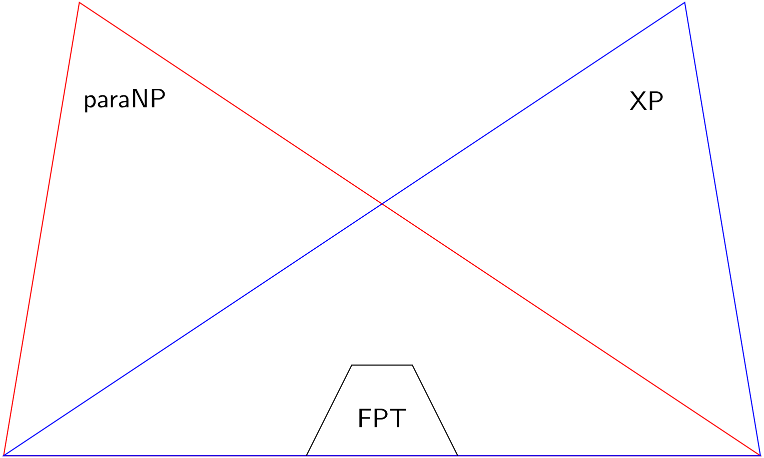
p-Exp-DTM-Halt
Input: A deterministic TM $\mathbb{M}$, $x \in \Sigma^*$ and an integer k,
Parameter: k
Question: Does $\mathbb{M}$ on input x stop in no more than $|x|^k$ steps?

Theorem

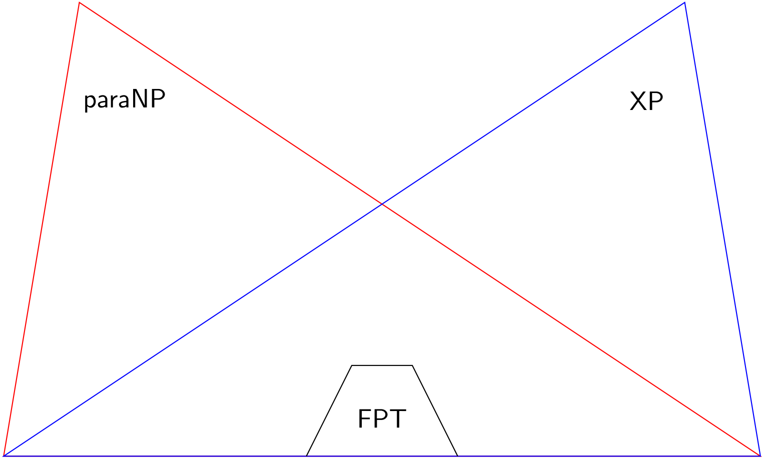
p-Exp-DTM-Halt *is XP-complete but does not belong to FPT.*

Relationships among classes

Relationships among classes



The W-hierarchy



Circuits: Depth and Weft

Circuits: Depth and Weft

- Let C be a boolean circuit: `AND` `OR` `NOT` gates.

Circuits: Depth and Weft

- Let C be a boolean circuit: AND OR NOT gates.
- A gate is small if it has only two or one input otherwise the gate is big

Circuits: Depth and Weft

- Let C be a boolean circuit: `AND` `OR` `NOT` gates.
- A gate is small if it has only two or one input otherwise the gate is big
- The depth of C is the maximum distance from an input gate to an output gate.

Circuits: Depth and Weft

- Let C be a boolean circuit: `AND` `OR` `NOT` gates.
- A gate is small if it has only two or one input otherwise the gate is big
- The depth of C is the maximum distance from an input gate to an output gate.
- The weft of C the maximum number of big gates in a path from an input gate to an output gate.

Circuits: Depth and Weft

- Let C be a boolean circuit: `AND` `OR` `NOT` gates.
- A gate is small if it has only two or one input otherwise the gate is big
- The depth of C is the maximum distance from an input gate to an output gate.
- The weft of C the maximum number of big gates in a path from an input gate to an output gate.
- Note that $\text{depth}(C) \geq \text{weft}(C)$

Variations on SAT

Variations on SAT

- The weight of an assignment $x = x_1 \dots x_n \in \{0,1\}^n$ is $W(x) = \sum_{i=1}^{n} x_i$; i.e., the number of ones

Variations on SAT

- The weight of an assignment $x = x_1 \dots x_n \in \{0,1\}^n$ is $W(x) = \sum_{i=1}^{n} x_i$; i.e., the number of ones
- A circuit C is k-satisfiable if there is a satisfying assignment with weight k.

Variations on SAT

- The weight of an assignment $x = x_1 \dots x_n \in \{0,1\}^n$ is $W(x) = \sum_{i=1}^{n} x_i$; i.e., the number of ones
- A circuit C is k-satisfiable if there is a satisfying assignment with weight k.
- A formula F is k-satisfiable if there is a satisfying assignment with weight k.

Variations on SAT

- The weight of an assignment $x = x_1 \ldots x_n \in \{0,1\}^n$ is $W(x) = \sum_{i=1}^{n} x_i$; i.e., the number of ones
- A circuit C is k-satisfiable if there is a satisfying assignment with weight k.
- A formula F is k-satisfiable if there is a satisfying assignment with weight k.

p-WSAT(FAM)
Input: A circuit/formula C/F in family FAM and an integer k,
Parameter: k
Question: Is C/F k-satisfiable?

W-classes

Families of circuits/formulas

W-classes

Families of circuits/formulas

- CIRC all boolean circuits

W-classes

Families of circuits/formulas

- CIRC all boolean circuits
- PROP all propositional formulas

W-classes

Families of circuits/formulas

- $\textsc{Circ}$ all boolean circuits
- $\textsc{Prop}$ all propositional formulas
- For $d \geq t \geq 0$, define

$$\mathcal{C}_{t,d} = \{c \mid C \in \textsc{Circ} \text{ and } \text{weft}(C) \leq t \text{ and } \text{depth}(C) \leq d\}$$

W-classes

Families of circuits/formulas

- CIRC all boolean circuits
- PROP all propositional formulas
- For $d \geq t \geq 0$, define

$$\mathcal{C}_{t,d} = \{c \mid C \in \textsc{Circ} \text{ and } \text{weft}(C) \leq t \text{ and } \text{depth}(C) \leq d\}$$

We define the following classes:

W-classes

Families of circuits/formulas

- $\textsc{Circ}$ all boolean circuits
- $\textsc{Prop}$ all propositional formulas
- For $d \geq t \geq 0$, define

$$\mathcal{C}_{t,d} = \{c \mid C \in \textsc{Circ} \text{ and } \mathrm{weft}(C) \leq t \text{ and } \mathrm{depth}(C) \leq d\}$$

We define the following classes:

- $W[P] = [\text{p-}\textsc{WSat}(\textsc{Circ})]^{fpt}$

W-classes

Families of circuits/formulas

- $\textsc{Circ}$ all boolean circuits
- $\textsc{Prop}$ all propositional formulas
- For $d \geq t \geq 0$, define

$$\mathcal{C}_{t,d} = \{c \mid C \in \textsc{Circ} \text{ and } \text{weft}(C) \leq t \text{ and } \text{depth}(C) \leq d\}$$

We define the following classes:

- $W[P] = [\text{p-}\textsc{WSat}(\textsc{Circ})]^{fpt}$
- $W[SAT] = [\text{p-}\textsc{WSat}(\textsc{Prop})]^{fpt}$

W-classes

Families of circuits/formulas

- $\textsc{Circ}$ all boolean circuits
- $\textsc{Prop}$ all propositional formulas
- For $d \geq t \geq 0$, define

$$\mathcal{C}_{t,d} = \{c \mid C \in \textsc{Circ} \text{ and } \text{weft}(C) \leq t \text{ and } \text{depth}(C) \leq d\}$$

We define the following classes:

- $W[P] = [\text{P-}\textsc{Wsat}(\textsc{Circ})]^{fpt}$
- $W[SAT] = [\text{P-}\textsc{Wsat}(\textsc{Prop})]^{fpt}$
- For $t \geq 1$, $W[t] = \{[\text{P-}\textsc{Wsat}(C_{t,d})]^{fpt} \mid d \geq 1\}$

W-hierarchy

- $W[P] = [\text{p-WSAT}(\text{CIRC})]^{fpt}$
- $W[SAT] = [\text{p-WSAT}(\text{PROP})]^{fpt}$
- For $t \geq 1$, $W[t] = \{[\text{p-WSAT}(C_{t,d})]^{fpt} \mid d \geq 1\}$

W-hierarchy

- $W[P] = [\text{p-WSAT}(\text{CIRC})]^{fpt}$
- $W[SAT] = [\text{p-WSAT}(\text{PROP})]^{fpt}$
- For $t \geq 1$, $W[t] = \{[\text{p-WSAT}(C_{t,d})]^{fpt} \mid d \geq 1\}$

Theorem

- $W[P] \subseteq paraNP \cap XP$
- $W[SAT] \subseteq W[P]$
- *For* $i \geq 1$, $W[i] \subseteq W[SAT]$ *and* $W[i] \subseteq W[i+1]$

W-hierarchy

W-hierarchy

Theorem

$FPT \subseteq W[1]$

W-hierarchy

Theorem

$FPT \subseteq W[1]$

Theorem

- *If, for some* $i \geq 1$, $FPT \neq W[i]$ *then* $P \neq NP$
- *If* $FPT \neq W[SAT]$ *then* $P \neq NP$
- *If* $FPT \neq W[P]$ *then* $P \neq NP$

W-hierarchy

Theorem

$FPT \subseteq W[1]$

Theorem

- *If, for some* $i \geq 1$, $FPT \neq W[i]$ *then* $P \neq NP$
- *If* $FPT \neq W[SAT]$ *then* $P \neq NP$
- *If* $FPT \neq W[P]$ *then* $P \neq NP$

Any of those conditions imply $FPT \neq paraNP$.

W-hierarchy

Theorem

$FPT \subseteq W[1]$

Theorem

- *If, for some* $i \geq 1$, $FPT \neq W[i]$ *then* $P \neq NP$
- *If* $FPT \neq W[SAT]$ *then* $P \neq NP$
- *If* $FPT \neq W[P]$ *then* $P \neq NP$

Any of those conditions imply $FPT \neq paraNP$.

Theorem

If $FPT = W[P]$ *then* CIRCUITSAT *for circuits with* n *inputs and* m *gates can be decided in* $2^{o(n)} m^{O(1)}$ *time.*

W[P]-hard problems

Some problems in $W[P]$

W[P]-hard problems

Some problems in $W[P]$

- p-Clique, p-DominantSet, p-SetCover

W[P]-hard problems

Some problems in $W[P]$

- p-Clique, p-DominantSet, p-SetCover

But in which level of the W-hierarchy?

W[P]-hard problems

Some problems in $W[P]$

- p-Clique, p-DominantSet, p-SetCover

But in which level of the W-hierarchy?

- p-Clique $\in W[1]$

W[P]-hard problems

Some problems in $W[P]$

- p-Clique, p-DominantSet, p-SetCover

But in which level of the W-hierarchy?

- p-Clique $\in W[1]$
  To prove this statement it is enough to show a circuit with weft 1 solving the problem (see blackboard)

W[P]-hard problems

Some problems in $W[P]$

- p-CLIQUE, p-DOMINANTSET, p-SETCOVER

But in which level of the W-hierarchy?

- p-CLIQUE $\in W[1]$
  To prove this statement it is enough to show a circuit with weft 1 solving the problem (see blackboard)
  In fact the problem is $W[1]$-complete

W[P]-hard problems

Some problems in $W[P]$

- p-Clique, p-DominantSet, p-SetCover

But in which level of the W-hierarchy?

- p-Clique $\in W[1]$
  To prove this statement it is enough to show a circuit with weft 1 solving the problem (see blackboard)
  In fact the problem is $W[1]$-complete
- p-Dominating Set $\in W[2]$ and p-SetCover $\in W[2]$ (Exercise)

W[P]-hard problems

Some problems in $W[P]$

- p-Clique, p-DominantSet, p-SetCover

But in which level of the W-hierarchy?

- p-Clique $\in W[1]$
  To prove this statement it is enough to show a circuit with weft 1 solving the problem (see blackboard)
  In fact the problem is $W[1]$-complete
- p-Dominating Set $\in W[2]$ and p-SetCover $\in W[2]$ (Exercise)
  In fact both problems are $W[2]$-complete

Exponential Time Hypothesis

Exponential Time Hypothesis

Exponential Time Hypothesis (ETH)

n-variable 3-SAT cannot be solved in time $2^{o(n)}$.

- We wish to get results like:

Exponential Time Hypothesis

Exponential Time Hypothesis (ETH)

n-variable 3-SAT cannot be solved in time $2^{o(n)}$.

- We wish to get results like:
  If there is an $f(k)\ n^{o(k)}$ time algorithm for problem XXX, then ETH fails.

Lower bounds for FPT algorithms

- We know that VERTEX COVER can be solved in time $O^*(c^k)$.

Lower bounds for FPT algorithms

- We know that VERTEX COVER can be solved in time $O^*(c^k)$.
- Can we do it much faster, for example in time $O^*(c^{\sqrt{k}})$ or $O^*(c^{k/logk})$?

Lemma

If VERTEX COVER can be solved in time $2^o(k)\ n^{O(1)}$, then ETH fails.

Lower bounds for FPT algorithms

- We know that VERTEX COVER can be solved in time $O^*(c^k)$.
- Can we do it much faster, for example in time $O^*(c^{\sqrt{k}})$ or $O^*(c^{k/logk})$?

Lemma

If VERTEX COVER can be solved in time $2^{o}(k)\ n^{O(1)}$, then ETH fails.

Proof.

There is a polynomial-time reduction from m-clause 3SAT to $O(m)$-vertex VERTEX COVER. The assumed algorithm would solve the latter problem in time $2^{o(m)}\ n^{O(1)}$, violating ETH. □

Efficient approximation schemes

- Polynomial-time approximation scheme (PTAS):
  Input: Instance $x, \epsilon > 0$
  Output: $(1 + \epsilon)$-approximate solution
  Running time: polynomial in $|x|$ for every fixed ϵ

Efficient approximation schemes

- Polynomial-time approximation scheme (PTAS):
  Input: Instance $x, \epsilon > 0$
  Output: $(1 + \epsilon)$-approximate solution
  Running time: polynomial in $|x|$ for every fixed ϵ
- PTAS: running time is $|x|^{f(1/\epsilon)}$

Efficient approximation schemes

- Polynomial-time approximation scheme (PTAS):
  Input: Instance $x, \epsilon > 0$
  Output: $(1+\epsilon)$-approximate solution
  Running time: polynomial in $|x|$ for every fixed ϵ
- PTAS: running time is $|x|^{f(1/\epsilon)}$
- Efficient PTAS (EPTAS)

Efficient approximation schemes

- Polynomial-time approximation scheme (PTAS):
  Input: Instance $x, \epsilon > 0$
  Output: $(1+\epsilon)$-approximate solution
  Running time: polynomial in $|x|$ for every fixed ϵ
- PTAS: running time is $|x|^{f(1/\epsilon)}$
- Efficient PTAS (EPTAS) running time is $f(1/\epsilon)|x|^{O(1)}$

Efficient approximation schemes

- Polynomial-time approximation scheme (PTAS):
  Input: Instance $x, \epsilon > 0$
  Output: $(1+\epsilon)$-approximate solution
  Running time: polynomial in $|x|$ for every fixed ϵ
- PTAS: running time is $|x|^{f(1/\epsilon)}$
- Efficient PTAS (EPTAS) running time is $f(1/\epsilon)|x|^{O(1)}$
- For some problems, there is a PTAS, but no EPTAS is known.

  Can we show that no EPTAS is possible?

No EPTAS?

No EPTAS?

Lemma

If the standard parameterization of an optimization problem is $W[1]$-hard, then there is no EPTAS for the optimization problem, unless $FPT = W[1]$.

No EPTAS?

Lemma

If the standard parameterization of an optimization problem is $W[1]$-hard, then there is no EPTAS for the optimization problem, unless $FPT = W[1]$.

Proof.

Suppose an $f(1/\epsilon)\ n^{O(1)}$ time EPTAS exists.
Running this EPTAS with $\epsilon = 1/(k+1)$ decides if the optimum is at most/at least k. □

Parameterized complexity

- Possibility to give evidence that certain problems are not FPT.
- Parameterized reduction.
- The W-hierarchy.
- ETH gives much stronger and tighter lower bounds.
- PTAS vs. EPTAS
- Kernel lower bounds