Algorithmic Graph Theory: How hard is your combinatorial optimization problem?

Arie M.C.A. Koster

Lecture 9

Clemson, June 13, 2017

CLEMSON UNIVERSITY
Lehrstuhl II für Mathematik
RWTH AACHEN UNIVERSITY

1 Parameterized Problems
2 W-Hierarchy
3 Designing Parameterized Algorithms

Definition

A parameterized problem is a pair (Π, κ), where Π is a decision problem with set of instances $\mathcal{I}$ and $\kappa : \mathcal{I} \to \mathbb{N}$ a so-called parameter, a in polynomial time (in the size of $\mathcal{I}$) computable function.

Definition

A parameterized problem is a pair (Π, κ), where Π is a decision problem with set of instances $\mathcal{I}$ and $\kappa : \mathcal{I} \to \mathbb{N}$ a so-called parameter, a in polynomial time (in the size of $\mathcal{I}$) computable function.

Parameterized problems are denoted by "p-" if parameterized by its "objective".

Example (p-VERTEX COVER)

Given: $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have a vertex cover of size at most k?

Definition

A parameterized problem is a pair (Π, κ), where Π is a decision problem with set of instances $\mathcal{I}$ and $\kappa : \mathcal{I} \to \mathbb{N}$ a so-called parameter, a in polynomial time (in the size of $\mathcal{I}$) computable function.

Parameterized problems are denoted by "p-" if parameterized by its "objective".

Example (p-VERTEX COVER)

Given: $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have a vertex cover of size at most k?

Example (p-tw-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: $tw(G)$
Question: Does G have an independent set of size at least k?

Parameterized Problem

Definition

A parameterized problem is a pair (Π, κ), where Π is a decision problem with set of instances $\mathcal{I}$ and $\kappa : \mathcal{I} \to \mathbb{N}$ a so-called parameter, a in polynomial time (in the size of $\mathcal{I}$) computable function.

Parameterized problems are denoted by "p-" if parameterized by its "objective".

Example (p-VERTEX COVER)

Given: $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have a vertex cover of size at most k?

Example (p-tw-INDEPENDENT SET)

Given: Graph $G = (V, E)$ of bounded treewidth and integer $k \in \mathbb{N}$
Parameter: $tw(G)$
Question: Does G have an independent set of size at least k?

Definition

Let (Π, κ) be a parameterized problem, $\mathcal{I}$ its set of instances.

- An algorithm A is called fixed parameter tractable (FPT) w.r.t. a parameter κ, if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial p, such that for every instance $I \in \mathcal{I}$, the running time of A is bounded by

$$f(\kappa(I)) \cdot p(|I|).$$

Definition

Let (Π, κ) be a parameterized problem, $\mathcal{I}$ its set of instances.

- An algorithm A is called fixed parameter tractable (FPT) w.r.t. a parameter κ, if there exists a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial p, such that for every instance $I \in \mathcal{I}$, the running time of A is bounded by
$$f(\kappa(I)) \cdot p(|I|).$$
- A parameterized problem (Π, κ) is called fixed parameter tractable (FPT) if there exists a FPT-algorithm w.r.t. κ solving the decision problem Π.

Definition

Let (Π, κ) be a parameterized problem, $\mathcal{I}$ its set of instances.

- An algorithm A is called fixed parameter tractable (FPT) w.r.t. a parameter κ, if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial p, such that for every instance $I \in \mathcal{I}$, the running time of A is bounded by
$$f(\kappa(I)) \cdot p(|I|).$$
- A parameterized problem (Π, κ) is called fixed parameter tractable (FPT) if there exists a FPT-algorithm w.r.t. κ solving the decision problem Π.
- $\mathcal{FPT}$ is the class of all parameterized problems that are fixed parameter tractable.

FPT Algorithm

Definition

Let (Π, κ) be a parameterized problem, $\mathcal{I}$ its set of instances.

- An algorithm A is called fixed parameter tractable (FPT) w.r.t. a parameter κ, if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial p, such that for every instance $I \in \mathcal{I}$, the running time of A is bounded by

$$f(\kappa(I)) \cdot p(|I|).$$

- A parameterized problem (Π, κ) is called fixed parameter tractable (FPT) if there exists a FPT-algorithm w.r.t. κ solving the decision problem Π.
- $\mathcal{FPT}$ is the class of all parameterized problems that are fixed parameter tractable.

Theorem

p-TREEWIDTH $\in \mathcal{FPT}$

Theorem

p-VERTEX COVER $\in \mathcal{FPT}$

Theorem

p-VERTEX COVER $\in \mathcal{FPT}$

Example (LOG-VERTEX COVER)

Given: $G = (V, E)$
Question: Does G have a vertex cover of size at most $\log |V|$?

Theorem

p-VERTEX COVER $\in \mathcal{FPT}$

Example (LOG-VERTEX COVER)

Given: $G = (V, E)$
Question: Does G have a vertex cover of size at most $\log |V|$?

Theorem

LOG-VERTEX COVER can be solved in $O(n^2)$

Theorem

p-VERTEX COVER $\in \mathcal{FPT}$

Example (LOG-VERTEX COVER)

Given: $G = (V, E)$
Question: Does G have a vertex cover of size at most $\log |V|$?

Theorem

LOG-VERTEX COVER can be solved in $O(n^2)$

Note: Every problem $\Pi \in \mathcal{P}$ is with every parameterization κ in $\mathcal{FPT}$.

Examples

Theorem

p-VERTEX COVER $\in \mathcal{FPT}$

Example (LOG-VERTEX COVER)

Given: $G = (V, E)$
Question: Does G have a vertex cover of size at most $\log |V|$?

Theorem

LOG-VERTEX COVER can be solved in $O(n^2)$

Note: Every problem $\Pi \in \mathcal{P}$ is with every parameterization κ in $\mathcal{FPT}$.
Note: Instead of multiplication, FPT can also be defined equivalently by

$$f(\kappa(I)) + p(|I|)$$

or the combination

$$g(\kappa(I)) + f(\kappa(I)) \cdot p(|I| + \kappa(I)).$$

How can we show a problem is in $\mathcal{FPT}$?

How can we show a problem is in $\mathcal{FPT}$?

Find a FPT-algorithm!

How can we show a problem is in $\mathcal{FPT}$?

Find a FPT-algorithm!

How can we show a problem is NOT in $\mathcal{FPT}$?

k-th Slice of a problem

How can we show a problem is in $\mathcal{FPT}$?

Find a FPT-algorithm!

How can we show a problem is NOT in $\mathcal{FPT}$?

Prove that no FPT-algorithm can exist, unless $\mathcal{P} = \mathcal{NP}$.

How can we show a problem is in $\mathcal{FPT}$?

Find a FPT-algorithm!

How can we show a problem is NOT in $\mathcal{FPT}$?

Prove that no FPT-algorithm can exist, unless $\mathcal{P} = \mathcal{NP}$.

Definition

Let (Π, κ) be a parameterized problem and $k \in \mathbb{N}$. Then, the *k-th slice* of (Π, κ) is the classical decision problem Π restricted to the instances I having $\kappa(I) = k$.

k-th Slice of a problem

How can we show a problem is in $\mathcal{FPT}$?

Find a FPT-algorithm!

How can we show a problem is NOT in $\mathcal{FPT}$?

Prove that no FPT-algorithm can exist, unless $\mathcal{P} = \mathcal{NP}$.

Definition

Let (Π, κ) be a parameterized problem and $k \in \mathbb{N}$. Then, the k-th slice of (Π, κ) is the classical decision problem Π restricted to the instances I having $\kappa(I) = k$.

Example (p-PARTITION IN INDEPENDENT SETS)

Given: $G = (V, E)$, integer $k \in \mathbb{N}$.
Parameter: k
Question: Does V have a partition in k independent sets?

Theorem

Let (Π, κ) *be a parameterized problem and* $k \in \mathbb{N}$. *Is* (Π, κ) *fixed parameter tractable, then the* k*-th slice* (Π, κ) *can be solved in polynomial time.*

k-th Slice of a problem

Theorem

Let (Π, κ) be a parameterized problem and $k \in \mathbb{N}$. Is (Π, κ) fixed parameter tractable, then the k-th slice (Π, κ) can be solved in polynomial time.

Corollary

Unless $\mathcal{P} = \mathcal{NP}$, p-PARTITION IN INDEPENDENT SETS$\notin \mathcal{FPT}$.

Theorem

Let (Π, κ) be a parameterized problem and $k \in \mathbb{N}$. Is (Π, κ) fixed parameter tractable, then the k-th slice (Π, κ) can be solved in polynomial time.

Corollary

Unless $\mathcal{P} = \mathcal{NP}$, p-PARTITION IN INDEPENDENT SETS$\notin \mathcal{FPT}$.

Note: If all slices can be solved in polynomial time, it is not yet clear that the problem is in $\mathcal{FPT}$.

Example (p-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have an independent set of size at least k?

Example (p-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$

Parameter: k

Question: Does G have an independent set of size at least k?

Algorithm `mis1`**(G).**

Input: Graph $G = (V, E)$.

Output: The maximum cardinality of an independent set of G.

if $|V| = 0$ **then**
  return 0

choose a vertex v of minimum degree in G

return $1 + \max\{\texttt{mis1}(G \setminus N[y]) : y \in N[v]\}$

Fig. 1.2 Algorithm `mis1` for MAXIMUM INDEPENDENT SET

Example (p-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have an independent set of size at least k?

Algorithm `mis1`**(G).**
Input: Graph $G = (V, E)$.
Output: The maximum cardinality of an independent set of G.

```
if |V| = 0 then
  └ return 0
choose a vertex v of minimum degree in G
return 1 + max{mis1(G \ N[y]) : y ∈ N[v]}
```

Fig. 1.2 Algorithm `mis1` for Maximum Independent Set

If k is added, a running time of $O((\Delta(G) + 1)^k n) = O(p(n))$ can be achieved (for fixed k).

Example (p-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have an independent set of size at least k?

Algorithm mis1(G).
Input: Graph $G = (V, E)$.
Output: The maximum cardinality of an independent set of G.

```
if |V| = 0 then
  return 0
choose a vertex v of minimum degree in G
return 1 + max{mis1(G \ N[y]) : y ∈ N[v]}
```

Fig. 1.2 Algorithm mis1 for MAXIMUM INDEPENDENT SET

If k is added, a running time of $O((\Delta(G)+1)^k n) = O(p(n))$ can be achieved (for fixed k).
This is not an FPT-algorithm! ($f(k)$ depends on $\Delta(G)$)

Example (p-deg-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: $k + \Delta(G)$
Question: Does G have an independent set of size at least k?

Example (cont.)

Example (p-deg-INDEPENDENT SET)

Given: Graph $G = (V, E)$ and integer $k \in \mathbb{N}$
Parameter: $k + \Delta(G)$
Question: Does G have an independent set of size at least k?

Corollary

p-deg-INDEPENDENT SET $\in \mathcal{FPT}$

For p-INDEPENDENT SET, the algorithm has running time $O(n^{k+1})$.

$\mathcal{XP}$-Algorithm

For p-INDEPENDENT SET, the algorithm has running time $O(n^{k+1})$.

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances.

- An algorithm A is called $\mathcal{XP}$-algorithm w.r.t. a parameterization $\kappa : \mathcal{I} \to \mathbb{N}$ if there exists computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that for every instance $I \in \mathcal{I}$ the running time of A is bounded by

$$f(\kappa(I)) \cdot |I|^{g(\kappa(I))}.$$

For p-INDEPENDENT SET, the algorithm has running time $O(n^{k+1})$.

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances.

- An algorithm A is called $\mathcal{XP}$-algorithm w.r.t. a parameterization $\kappa : \mathcal{I} \to \mathbb{N}$ if there exists computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that for every instance $I \in \mathcal{I}$ the running time of A is bounded by

$$f(\kappa(I)) \cdot |I|^{g(\kappa(I))}.$$

- $\mathcal{XP}$ defines the set of all parameterized problems for which an $\mathcal{XP}$-algorithm exists.

$\mathcal{XP}$-Algorithm

For p-INDEPENDENT SET, the algorithm has running time $O(n^{k+1})$.

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances.

- An algorithm A is called $\mathcal{XP}$-algorithm w.r.t. a parameterization $\kappa : \mathcal{I} \to \mathbb{N}$ if there exists computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that for every instance $I \in \mathcal{I}$ the running time of A is bounded by

$$f(\kappa(I)) \cdot |I|^{g(\kappa(I))}.$$

- $\mathcal{XP}$ defines the set of all parameterized problems for which an $\mathcal{XP}$-algorithm exists.

Note: $\mathcal{FPT} \subseteq \mathcal{XP}$

$\mathcal{XP}$-Algorithm

For p-INDEPENDENT SET, the algorithm has running time $O(n^{k+1})$.

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances.

- An algorithm A is called $\mathcal{XP}$-algorithm w.r.t. a parameterization $\kappa : \mathcal{I} \to \mathbb{N}$ if there exists computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that for every instance $I \in \mathcal{I}$ the running time of A is bounded by

$$f(\kappa(I)) \cdot |I|^{g(\kappa(I))}.$$

- $\mathcal{XP}$ defines the set of all parameterized problems for which an $\mathcal{XP}$-algorithm exists.

Note: $\mathcal{FPT} \subseteq \mathcal{XP}$

Theorem

p-INDEPENDENT SET $\in \mathcal{XP}$

1 Parameterized Problems
2 W-Hierarchy
3 Designing Parameterized Algorithms

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Yes, we can!

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Definition

A parameterized problem (Π_1, κ_1) reduces parameterized to a parameterized problem (Π_2, κ_2) if there exists a function $r : \mathcal{I}_1 \to \mathcal{I}_2$ such that

- for all $I \in \mathcal{I}_1$, I is a "yes"-instance of Π_1 if and only if $r(I)$ is a "yes"-instance of Π_2.

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Definition

A parameterized problem (Π_1, κ_1) reduces parameterized to a parameterized problem (Π_2, κ_2) if there exists a function $r : \mathcal{I}_1 \to \mathcal{I}_2$ such that

- for all $I \in \mathcal{I}_1$, I is a "yes"-instance of Π_1 if and only if $r(I)$ is a "yes"-instance of Π_2.
- $r(I)$ is computable in time $f(\kappa_1(I)) \cdot p(|I|)$ for a computable function f and a polynomial p.

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Definition

A parameterized problem (Π_1, κ_1) reduces parameterized to a parameterized problem (Π_2, κ_2) if there exists a function $r : \mathcal{I}_1 \to \mathcal{I}_2$ such that

- for all $I \in \mathcal{I}_1$, I is a "yes"-instance of Π_1 if and only if $r(I)$ is a "yes"-instance of Π_2.
- $r(I)$ is computable in time $f(\kappa_1(I)) \cdot p(|I|)$ for a computable function f and a polynomial p.
- for $I \in \mathcal{I}_1$ and a computable function $g : \mathbb{N} \to \mathbb{N}$ it holds $\kappa_2(r(I)) \leq g(\kappa_1(I))$.

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Definition

A parameterized problem (Π_1, κ_1) reduces parameterized to a parameterized problem (Π_2, κ_2) if there exists a function $r : \mathcal{I}_1 \to \mathcal{I}_2$ such that

- for all $I \in \mathcal{I}_1$, I is a "yes"-instance of Π_1 if and only if $r(I)$ is a "yes"-instance of Π_2.
- $r(I)$ is computable in time $f(\kappa_1(I)) \cdot p(|I|)$ for a computable function f and a polynomial p.
- for $I \in \mathcal{I}_1$ and a computable function $g : \mathbb{N} \to \mathbb{N}$ it holds $\kappa_2(r(I)) \leq g(\kappa_1(I))$.

- The classical reduction of p-INDEPENDENT SET to p-VERTEX COVER is not a parameterized reduction

Can we distinguish between $\mathcal{FPT}$ and $\mathcal{XP}$ in more detail?

Definition

A parameterized problem (Π_1, κ_1) reduces parameterized to a parameterized problem (Π_2, κ_2) if there exists a function $r : \mathcal{I}_1 \to \mathcal{I}_2$ such that

- for all $I \in \mathcal{I}_1$, I is a "yes"-instance of Π_1 if and only if $r(I)$ is a "yes"-instance of Π_2.
- $r(I)$ is computable in time $f(\kappa_1(I)) \cdot p(|I|)$ for a computable function f and a polynomial p.
- for $I \in \mathcal{I}_1$ and a computable function $g : \mathbb{N} \to \mathbb{N}$ it holds $\kappa_2(r(I)) \leq g(\kappa_1(I))$.

- The classical reduction of p-INDEPENDENT SET to p-VERTEX COVER is not a parameterized reduction
- p-INDEPENDENT SET reduces parameterized to p-CLIQUE

Theorem

Let (Π_1, κ_1) *and* (Π_2, κ_2) *be parameterized problems. If* (Π_1, κ_1) *reduces parameterized to* (Π_2, κ_2)*, and* $(\Pi_2, \kappa_2) \in \mathcal{FPT}$*, then* $(\Pi_1, \kappa_1) \in \mathcal{FPT}$*.*

Theorem

Let (Π_1, κ_1) and (Π_2, κ_2) be parameterized problems. If (Π_1, κ_1) reduces parameterized to (Π_2, κ_2), and $(\Pi_2, \kappa_2) \in \mathcal{FPT}$, then $(\Pi_1, \kappa_1) \in \mathcal{FPT}$.

SAT is the classical $\mathcal{NP}$-complete problem. Can we define something similar for parameterized complexity?

SAT is the classical $\mathcal{NP}$-complete problem. Can we define something similar for parameterized complexity?

Definition (p-WEIGHTED SAT)

Given: a boolean formula and integer $k \in \mathbb{N}$
Parameter: k
Question: Is the boolean formula satisfiable with at least k variables set to TRUE?

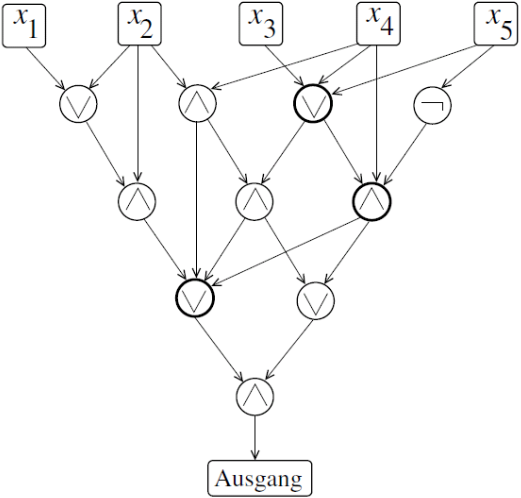
SAT is the classical $\mathcal{NP}$-complete problem. Can we define something similar for parameterized complexity?

Definition (p-WEIGHTED SAT)

Given: a boolean formula and integer $k \in \mathbb{N}$
Parameter: k
Question: Is the boolean formula satisfiable with at least k variables set to TRUE?

Definition (WEIGHTED WEFT-t-DEPTH-d SAT)

Given: A Boolean formula of depth at most d and weft at most t, and a number k. The depth is the maximal number of gates on any path from the root to a leaf, and the weft is the maximal number of gates of fan-in at least three on any path from the root to a leaf.
Question: Is the boolean formula satisfiable with k variables set to TRUE?



Boolean circuit with weft=3 and depth=5

Es is conjuctured that boolean formula with high weft are more difficult than those with low weft.

W-Hierarchy

Es is conjuctured that boolean formula with high weft are more difficult than those with low weft.

Definition

The W-Hierarchy consists of the complexity classes $W[t]$, $t \geq 1$. A parameterized problem (Π, κ) is a member of $W[t]$ if it can be reduced parameterized to p-WEIGHTED WEFT-t-DEPTH-d SAT for some $d \in \mathbb{N}$.

Es is conjuctured that boolean formula with high weft are more difficult than those with low weft.

Definition

The W-Hierarchy consists of the complexity classes $W[t]$, $t \geq 1$. A parameterized problem (Π, κ) is a member of $W[t]$ if it can be reduced parameterized to p-WEIGHTED WEFT-t-DEPTH-d SAT for some $d \in \mathbb{N}$.

Example

p-INDEPENDENT SET$\in W[1]$

Es is conjuctured that boolean formula with high weft are more difficult than those with low weft.

Definition

The W-Hierarchy consists of the complexity classes $W[t]$, $t \geq 1$. A parameterized problem (Π, κ) is a member of $W[t]$ if it can be reduced parameterized to p-WEIGHTED WEFT-t-DEPTH-d SAT for some $d \in \mathbb{N}$.

Example

p-INDEPENDENT SET$\in W[1]$

Example

p-CLIQUE$\in W[1]$

Example (p-DOMINATING SET)

Given: Graph $G = (V, E)$, integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have a dominating set of size at most k, i.e., a subset of the vertices $S \subseteq V$ such that for all vertices $v \in V$: $N[v] \cap S \neq \emptyset$.

Example (p-DOMINATING SET)

Given: Graph $G = (V, E)$, integer $k \in \mathbb{N}$
Parameter: k
Question: Does G have a dominating set of size at most k, i.e., a subset of the vertices $S \subseteq V$ such that for all vertices $v \in V$: $N[v] \cap S \neq \emptyset$.

Lemma

p-DOMINATING SET $\in W[2]$

Definition

- A parameterized problem (Π, κ) is $W[t]$-hard if every problem in $W[t]$ can be reduced parameterized to (Π, κ).

Definition

- A parameterized problem (Π, κ) is $W[t]$-hard if every problem in $W[t]$ can be reduced parameterized to (Π, κ).
- A parameterized problem (Π, κ) is $W[t]$-complete if it is $W[t]$-hard and a member of $W[t]$ itself.

Corollary: p-WEIGHTED WEFT-t-DEPTH-d SAT is $W[t]$-complete (by definition).

Definition

- A parameterized problem (Π, κ) is $W[t]$-hard if every problem in $W[t]$ can be reduced parameterized to (Π, κ).
- A parameterized problem (Π, κ) is $W[t]$-complete if it is $W[t]$-hard and a member of $W[t]$ itself.

Corollary: p-WEIGHTED WEFT-t-DEPTH-d SAT is $W[t]$-complete (by definition).

Theorem

- *p-INDEPENDENT SET and p-CLIQUE are $W[1]$-complete*

$W[t]$-Completeness

Definition

- A parameterized problem (Π, κ) is $W[t]$-hard if every problem in $W[t]$ can be reduced parameterized to (Π, κ).
- A parameterized problem (Π, κ) is $W[t]$-complete if it is $W[t]$-hard and a member of $W[t]$ itself.

Corollary: p-WEIGHTED WEFT-t-DEPTH-d SAT is $W[t]$-complete (by definition).

Theorem

- *p-INDEPENDENT SET and p-CLIQUE are $W[1]$-complete*
- *p-DOMINATING SET is $W[2]$-complete*

Theorem

For every $t \geq 1$, $W[t] = \mathcal{FPT}$ if and only if a $W[t]$-hard problem is a member of $\mathcal{FPT}$.

1 Parameterized Problems
2 W-Hierarchy
3 Designing Parameterized Algorithms

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances of Π. A in polynomial time computable function $f : \mathcal{I} \times \mathbb{N} \to \mathcal{I} \times \mathbb{N}$ is called kernelization for (Π, κ) if $(I', \kappa(I')) = f(I, \kappa(I))$ satisfies the following three properties:

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances of Π. A in polynomial time computable function $f : \mathcal{I} \times \mathbb{N} \to \mathcal{I} \times \mathbb{N}$ is called kernelization for (Π, κ) if $(I', \kappa(I')) = f(I, \kappa(I))$ satisfies the following three properties:

1. For all $I \in \mathcal{I}$, $(I, \kappa(I))$ is a "yes"-instance if and only if $(I', \kappa(I'))$ a "yes"-instance of Π

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances of Π. A in polynomial time computable function $f : \mathcal{I} \times \mathbb{N} \to \mathcal{I} \times \mathbb{N}$ is called kernelization for (Π, κ) if $(I', \kappa(I')) = f(I, \kappa(I))$ satisfies the following three properties:

1. For all $I \in \mathcal{I}$, $(I, \kappa(I))$ is a "yes"-instance if and only if $(I', \kappa(I'))$ a "yes"-instance of Π
2. There exists a function $f' : \mathbb{N} \to \mathbb{N}$ such that $|I'| \leq f'(\kappa(I))$

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances of Π. A in polynomial time computable function $f : \mathcal{I} \times \mathbb{N} \to \mathcal{I} \times \mathbb{N}$ is called kernelization for (Π, κ) if $(I', \kappa(I')) = f(I, \kappa(I))$ satisfies the following three properties:

1. For all $I \in \mathcal{I}$, $(I, \kappa(I))$ is a "yes"-instance if and only if $(I', \kappa(I'))$ a "yes"-instance of Π
2. There exists a function $f' : \mathbb{N} \to \mathbb{N}$ such that $|I'| \leq f'(\kappa(I))$
3. $\kappa(I') \leq \kappa(I)$

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances of Π. A in polynomial time computable function $f : \mathcal{I} \times \mathbb{N} \to \mathcal{I} \times \mathbb{N}$ is called kernelization for (Π, κ) if $(I', \kappa(I')) = f(I, \kappa(I))$ satisfies the following three properties:

1. For all $I \in \mathcal{I}$, $(I, \kappa(I))$ is a "yes"-instance if and only if $(I', \kappa(I'))$ a "yes"-instance of Π
2. There exists a function $f' : \mathbb{N} \to \mathbb{N}$ such that $|I'| \leq f'(\kappa(I))$
3. $\kappa(I') \leq \kappa(I)$

The instance I' is called kernel of (Π, κ) and $f'(\kappa(I))$ is called the size of the kernel.

Idea: reduce an instance I to an instance I' which size only depends on the parameter, not on the original instance size

Definition

Let (Π, κ) be a parameterized problem with $\mathcal{I}$ the set of instances of Π. A in polynomial time computable function $f : \mathcal{I} \times \mathbb{N} \to \mathcal{I} \times \mathbb{N}$ is called kernelization for (Π, κ) if $(I', \kappa(I')) = f(I, \kappa(I))$ satisfies the following three properties:

1. For all $I \in \mathcal{I}$, $(I, \kappa(I))$ is a "yes"-instance if and only if $(I', \kappa(I'))$ a "yes"-instance of Π
2. There exists a function $f' : \mathbb{N} \to \mathbb{N}$ such that $|I'| \leq f'(\kappa(I))$
3. $\kappa(I') \leq \kappa(I)$

The instance I' is called kernel of (Π, κ) and $f'(\kappa(I))$ is called the size of the kernel.

Example: p-VERTEX COVER

2 Reduction rules:

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree 0. G has a vertex cover of size k, if and only if $G - v$ has a vertex cover of size k.

2 Reduction rules:

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree 0. G has a vertex cover of size k, if and only if $G - v$ has a vertex cover of size k.

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree at least $k + 1$. G has a vertex cover of size k if and only if $G - v$ has a vertex cover of size $k - 1$.

2 Reduction rules:

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree 0. G has a vertex cover of size k, if and only if $G - v$ has a vertex cover of size k.

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree at least $k + 1$. G has a vertex cover of size k if and only if $G - v$ has a vertex cover of size $k - 1$.

Lemma

Let $G = (V, E)$ be a graph without isolated vertices. If G has a vertex cover of size at most k and $\Delta(G) \leq d$, then G has at most $k(d + 1)$ vertices.

Lemma

p-VERTEX COVER has a kernel of size at most $k(k+1)$, where k is the parameter of the problem.

Lemma

p-VERTEX COVER has a kernel of size at most $k(k+1)$, where k is the parameter of the problem.

Theorem

- *If a parameterized problem (Π, κ) has a kernelization, then the problem is a member of $\mathcal{FPT}$*

Lemma

p-VERTEX COVER has a kernel of size at most $k(k+1)$, where k is the parameter of the problem.

Theorem

- *If a parameterized problem (Π, κ) has a kernelization, then the problem is a member of $\mathcal{FPT}$*
- *Every problem (Π, κ) in the class $\mathcal{FPT}$ can be solved by a kernelization algorithm.*

Lemma

p-VERTEX COVER has a kernel of size at most $k(k+1)$, where k is the parameter of the problem.

Theorem

- *If a parameterized problem (Π, κ) has a kernelization, then the problem is a member of $\mathcal{FPT}$*
- *Every problem (Π, κ) in the class $\mathcal{FPT}$ can be solved by a kernelization algorithm.*

Corollary

p-VERTEX COVER$\in$ FPT

FPT algorithm for p-VERTEX COVER

1. $(G,k) \to (G-v, k-1)$ for all vertices v of degree $> k$

FPT algorithm for p-VERTEX COVER

1. $(G, k) \to (G - v, k - 1)$ for all vertices v of degree $> k$
2. remove isolated vertices

FPT algorithm for p-VERTEX COVER

1. $(G, k) \to (G - v, k - 1)$ for all vertices v of degree $> k$
2. remove isolated vertices
3. If $|V| > k(k+1)$, then **return** "No"

FPT algorithm for p-VERTEX COVER

1. $(G, k) \to (G - v, k - 1)$ for all vertices v of degree $> k$
2. remove isolated vertices
3. If $|V| > k(k+1)$, then **return** "No"
4. If $|V| \leq k(k+1)$, then enumerate all subsets S, and check on vertex cover, take the smallest.
   If $|S| \leq k$, then **return** "Yes" else **return** "No"

FPT algorithm for p-VERTEX COVER

1. $(G, k) \to (G - v, k - 1)$ for all vertices v of degree $> k$
2. remove isolated vertices
3. If $|V| > k(k+1)$, then **return** "No"
4. If $|V| \leq k(k+1)$, then enumerate all subsets S, and check on vertex cover, take the smallest.
   If $|S| \leq k$, then **return** "Yes" else **return** "No"

Theorem

The FPT-algorithm for p-VERTEX COVER decides for every graph G with n vertices in $O(kn + k^{2k})$ whether G has a vertex cover of size at most k.

Lemma

Let $G = (V, E)$ *be a graph and* $v \in V$ *a vertex of degree 1 with neighbor* w*.* G *has a vertex cover of size* k *if and only if* $G - \{v, w\}$ *has a vertex cover of size* $k - 1$*.*

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree 1 with neighbor w. G has a vertex cover of size k if and only if $G - \{v, w\}$ has a vertex cover of size $k - 1$.

Theorem

p-VERTEX COVER has a kernel of size at most k^2

Lemma

Let $G = (V, E)$ be a graph and $v \in V$ a vertex of degree 1 with neighbor w. G has a vertex cover of size k if and only if $G - \{v, w\}$ has a vertex cover of size $k - 1$.

Theorem

p-VERTEX COVER has a kernel of size at most k^2

Theorem

p-VERTEX COVER has a kernel of size at most $2k$

Search trees of limited height: Example p-VERTEX COVER (earlier)

Search trees of limited height: Example p-VERTEX COVER (earlier)

3 rules:

VC1 If v is a vertex of degree 1, add $w \in N(v)$ to the vertex cover; continue with $(G - w, k - 1)$

Search trees of limited height: Example p-VERTEX COVER (earlier)

3 rules:

- VC1 If v is a vertex of degree 1, add $w \in N(v)$ to the vertex cover; continue with $(G - w, k - 1)$
- VC2 If v is a vertex of degree 2, either both neighbors of v are part of the vertex cover or v and all neighbors of both neighbors of v are: branch into $(G - N(v), k - 2)$ and $(G - v - N_2(v), k - \ell)$

Search trees of limited height: Example p-VERTEX COVER (earlier)
3 rules:

- VC1 If v is a vertex of degree 1, add $w \in N(v)$ to the vertex cover; continue with $(G - w, k - 1)$
- VC2 If v is a vertex of degree 2, either both neighbors of v are part of the vertex cover or v and all neighbors of both neighbors of v are: branch into $(G - N(v), k - 2)$ and $(G - v - N_2(v), k - \ell)$
- VC3 If v is a vertex of degree at least 3, then either v or all its neighbors are part of the vertex cover: branch into $(G - v, k - 1)$ and $(G - N(v), k - |N(v)|)$.

Search trees of limited height: Example p-VERTEX COVER (earlier)
3 rules:

- VC1 If v is a vertex of degree 1, add $w \in N(v)$ to the vertex cover; continue with $(G - w, k - 1)$
- VC2 If v is a vertex of degree 2, either both neighbors of v are part of the vertex cover or v and all neighbors of both neighbors of v are: branch into $(G - N(v), k - 2)$ and $(G - v - N_2(v), k - \ell)$
- VC3 If v is a vertex of degree at least 3, then either v or all its neighbors are part of the vertex cover: branch into $(G - v, k - 1)$ and $(G - N(v), k - |N(v)|)$.

Theorem

The search tree defined by VC1, VC2, and VC3 for p-VERTEX COVER has a size of $O(1.47^k)$.

Search trees of limited height: Example p-VERTEX COVER (earlier)

3 rules:

- VC1 If v is a vertex of degree 1, add $w \in N(v)$ to the vertex cover; continue with $(G - w, k - 1)$
- VC2 If v is a vertex of degree 2, either both neighbors of v are part of the vertex cover or v and all neighbors of both neighbors of v are: branch into $(G - N(v), k - 2)$ and $(G - v - N_2(v), k - \ell)$
- VC3 If v is a vertex of degree at least 3, then either v or all its neighbors are part of the vertex cover: branch into $(G - v, k - 1)$ and $(G - N(v), k - |N(v)|)$.

Theorem

The search tree defined by VC1, VC2, and VC3 for p-VERTEX COVER has a size of $O(1.47^k)$.

A combination of kernelization and search tree is also possible.

Algorithmic Graph Theory: How hard is your combinatorial optimization problem?

Arie M.C.A. Koster

Lecture 9

Clemson, June 13, 2017

Clemson University
Lehrstuhl II für Mathematik
RWTH Aachen University