Randomized Methods in Parameterized Algorithms

Based on *Parameterized Algorithms* by Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk and Saket Saurabh

Alek Westover, Tomasz Ślusarczyk, Sarah Zhao

February 2, 2024

Introduction

Parameterization

We can parameterize a problem by assigning k to each input instance x.

Introduction

Parameterization

We can parameterize a problem by assigning k to each input instance x.

Motivating Example: Min Vertex Cover is NP-Hard

Introduction

Parameterization

We can parameterize a problem by assigning k to each input instance x.

Motivating Example: Min Vertex Cover is NP-Hard

k-Vertex Cover is P, for small k.

Fixed-Parameter Tractable

Examples of possible parameters:

(a) treewidth
(b) number of vertices/edges in solution
(c) max degree of input graph
(d) number of clauses in CNF
(e) size of alphabet

Fixed-parameter Tractable (FPT)

A problem is FPT if there exists some algorithm with time $f(k)n^{\mathcal{O}(1)}$, for computable f and n is polynomial for some constant independent of n and k.

Kernelization

Lemma

*A parameterized problem is FPT (***fixed parameter tractable***) if and only if it admits a kernel.*

Kernelization

Lemma

*A parameterized problem is FPT (***fixed parameter tractable***) if and only if it admits a kernel.*

Kernelization

Given an instance (I, k) of parameterized problem Q, map to an equivalent (I', k'), such that

(a) $k' \leq k$,

(b) (I, k) is yes-instance $\iff$ (I', k') is yes-instance,

(c) $|I'| \leq f(k)$ for some f.

Vertex Cover

Problem: Given graph G with n vertices, find a set of k vertices that includes at least one endpoint of every edge in $E(G)$.
Reduction:

Vertex Cover

Problem: Given graph G with n vertices, find a set of k vertices that includes at least one endpoint of every edge in $E(G)$.
Reduction:

1. Remove all isolated vertices from G. $(G, k) \leftarrow (G - v, k)$

Vertex Cover

Problem: Given graph G with n vertices, find a set of k vertices that includes at least one endpoint of every edge in $E(G)$.
Reduction:

1. Remove all isolated vertices from G. $(G, k) \leftarrow (G - v, k)$
2. If v has degree $\geq k+1$, remove v and incident edges. $(G, k) \leftarrow (G - v, k - 1)$

Vertex Cover

Problem: Given graph G with n vertices, find a set of k vertices that includes at least one endpoint of every edge in $E(G)$.
Reduction:

1. Remove all isolated vertices from G. $(G, k) \leftarrow (G - v, k)$
2. If v has degree $\geq k + 1$, remove v and incident edges. $(G, k) \leftarrow (G - v, k - 1)$
3. If $k < 0$ and $|V(G)| > k^2 + k$ or $|E(G)| > k^2$, then no-instance. Else, yes.

Vertex Cover

Problem: Given graph G with n vertices, find a set of k vertices that includes at least one endpoint of every edge in $E(G)$.
Reduction:

1. Remove all isolated vertices from G. $(G, k) \leftarrow (G - v, k)$
2. If v has degree $\geq k + 1$, remove v and incident edges. $(G, k) \leftarrow (G - v, k - 1)$
3. If $k < 0$ and $|V(G)| > k^2 + k$ or $|E(G)| > k^2$, then no-instance. Else, yes.

If G is yes-instance and cannot perform 1) or 2), then $V(G) \geq k^2 + k$ and $|E(g)| \geq k^2$.

Kernel Size: $\mathcal{O}(k^2)$ vertices and edges

Randomized Methods

Plan:

1. Use randomness to *highlight* the solution if we get **lucky**.
2. Simple DP to find a highlighted solution.
3. Repeat to amplify success probability.

Fact

Suppose event X occurs with probability p. Then, if we perform $1/p$ independent trials of X the probability that at least one succeeds is $\Omega(1)$.

Proof.

$(1-p)^{1/p} \approx 1/e$. □

Hope: $\Pr[\text{lucky}] \geq \frac{1}{f(k)n^{O(1)}}$.

Longest Path

Color vertices with k colors.
Hope: *rainbow path* (all k vertices get distinct colors).
Finding a rainbow path: Dynamic programming.
For each $v \in V, C \subseteq [k]$:

$$\text{good}[v, C] = \text{“is there a path using exactly colors } C \text{ ending at vertex } v\text{?”}$$

Compute the DP in order of increasing $|C|$.

$$\text{good}[v, C] = \text{“Does } v \text{ have a neighbor } w \text{ such that good}[w, C \setminus \{c\}]\text{?”}$$

Longest Path

Pr[coloring succeeds] $\approx e^{-k}$

States: $2^k \cdot n$

Operations when propogating DP:

$$\sum_{i=1}^{k}(m+n)\binom{k}{i} \le (m+n)2^k.$$

e^k tries, indexing into array: k.

Fact

All graphs with more than kn edges contain a k-path.

Theorem

There is a randomized algorithm for k-path with running time

$$\mathcal{O}(nk^2(2e)^k)$$

using space $O(2^k n)$.

Subgraph Isomorphism

Problem (Subgraph Isomorphism)

For given graph G with n vertices and H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Subgraph Isomorphism

Problem (Subgraph Isomorphism)

For given graph G with n vertices and H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Question

Is Subgraph Isomorphism *FPT?*

Subgraph Isomorphism

Problem (Subgraph Isomorphism)

For given graph G with n vertices and H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Question

Is Subgraph Isomorphism *FPT?*

Answer

*Probably not. Already a special case for $H = K_k$ (k-*Clique*) is believed not to be FPT.*

Subgraph Isomorphism

Problem (Subgraph Isomorphism)

For given graph G with n vertices and H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Question

Is Subgraph Isomorphism FPT parametrized by k?

Answer

Probably not. Already a special case for $H = K_k$ (k-Clique) is believed not to be FPT.

Subgraph Isomorphism

Problem (Subgraph Isomorphism)

For given graph G with $\Delta(G) \leq d$ with n vertices and H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Question

Is Subgraph Isomorphism *FPT parametrized by k and d?*

Subgraph Isomorphism

Problem (Subgraph Isomorphism)

For given graph G with $\Delta(G) \leq d$ with n vertices and H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Question

Is Subgraph Isomorphism *FPT parametrized by k and d?*

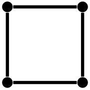
Answer

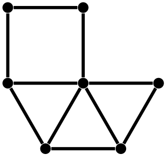
Yes! We will show it for connected H for technical reasons.

Subgraph Isomorphism – random FPT algorithm

Problem (SI)

For given graph G with $\Delta(G) \leq d$ with n vertices and connected H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?



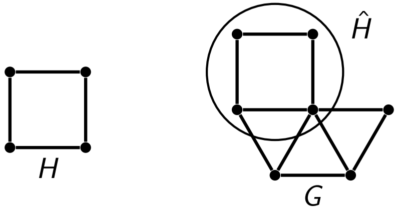


Subgraph Isomorphism – random FPT algorithm

Problem (SI)

For given graph G with $\Delta(G) \leq d$ with n vertices and connected H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Fix $\hat{H} \subseteq G$ such that $\hat{H} \cong H$ (if it doesn't exist, our algorithm will surely return NO).

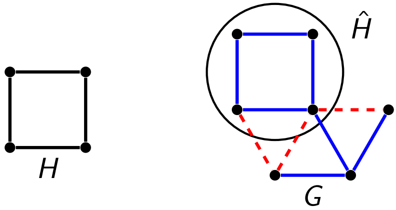


Subgraph Isomorphism – random FPT algorithm

Problem (SI)

For given graph G with $\Delta(G) \leq d$ with n vertices and connected H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Fix $\hat{H} \subseteq G$ such that $\hat{H} \cong H$ (if it doesn't exist, our algorithm will surely return NO).Color all edges of G red and blue, independently and uniformly at random.



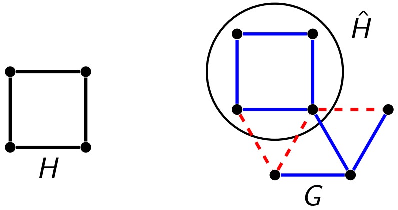
Subgraph Isomorphism – random FPT algorithm

Problem (SI)

For given graph G with $\Delta(G) \leq d$ with n vertices and connected H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Fix $\hat{H} \subseteq G$ such that $\hat{H} \cong H$ (if it doesn't exist, our algorithm will surely return NO).Color all edges of G red and blue, independently and uniformly at random.

$$\mathcal{L} = \hat{H} \text{ is blue} \wedge (\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})$$



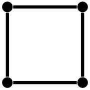
SUBGRAPH ISOMORPHISM – random FPT algorithm

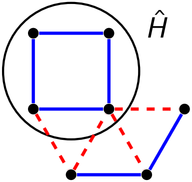
Problem (SI)

For given graph G with $\Delta(G) \leq d$ with n vertices and connected H with at most k vertices, is there a $\hat{H} \subseteq G$ such that $\hat{H} \cong H$?

Fix $\hat{H} \subseteq G$ such that $\hat{H} \cong H$ (if it doesn't exist, our algorithm will surely return NO).Color all edges of G red and blue, independently and uniformly at random.

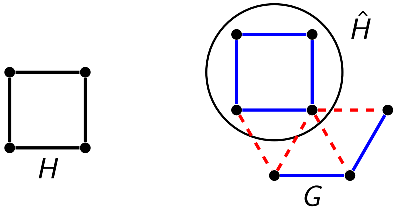
$$\mathcal{L} = \hat{H} \text{ is blue} \wedge (\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})$$





Subgraph Isomorphism – random FPT algorithm

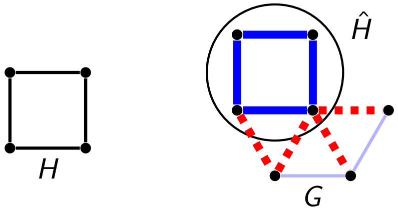
$$\mathcal{L} = \hat{H} \text{ is blue} \wedge (\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})$$



Conditioning on $\mathcal{L}$, $\hat{H}$ is a connected component of the blue subgraph of G. We can find it by checking all components in time $nk!k^{\mathcal{O}(1)}$.

Subgraph Isomorphism – random FPT algorithm

$$\mathcal{L} = \hat{H} \text{ is blue} \wedge (\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})$$



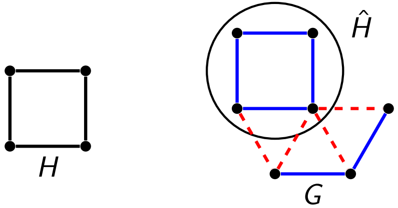
Conditioning on $\mathcal{L}$, $\hat{H}$ is a connected component of the blue subgraph of G. We can find it by checking all components in time $nk!k^{\mathcal{O}(1)}$.

$$\mathbb{P}[\mathcal{L}] = \mathbb{P}[\hat{H} \text{ is blue}] \cdot \mathbb{P}[\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})] \geq 2^{-kd}$$

since we fix color of at most kd edges.

SUBGRAPH ISOMORPHISM – random FPT algorithm

$$\mathcal{L} = \hat{H} \text{ is blue} \wedge (\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})$$



Conditioning on $\mathcal{L}$, $\hat{H}$ is a connected component of the blue subgraph of G. We can find it by checking all components in time $nk!k^{\mathcal{O}(1)}$.

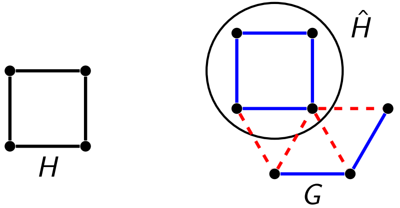
$$\mathbb{P}[\mathcal{L}] = \mathbb{P}[\hat{H} \text{ is blue}] \cdot \mathbb{P}[\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})] \geq 2^{-kd}$$

since we fix color of at most kd edges.

Total runtime: $k!k^{\mathcal{O}(1)}2^{kd}\mathcal{O}(n)$ (can be improved to $k^{\mathcal{O}(d \log d)}2^{kd}\mathcal{O}(n)$).

Subgraph Isomorphism – random FPT algorithm

$$\mathcal{L} = \hat{H} \text{ is blue} \wedge (\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})$$



Conditioning on $\mathcal{L}$, $\hat{H}$ is a connected component of the blue subgraph of G. We can find it by checking all components in time $nk!k^{\mathcal{O}(1)}$.

$$\mathbb{P}[\mathcal{L}] = \mathbb{P}[\hat{H} \text{ is blue}] \cdot \mathbb{P}[\forall v \in \hat{H}, u \notin \hat{H} : uv \text{ is red})] \geq 2^{-kd}$$

since we fix color of at most kd edges.

Total runtime: $k!k^{\mathcal{O}(1)}2^{kd}\mathcal{O}(n)$ (can be improved to $k^{\mathcal{O}(d \log d)}2^{kd}\mathcal{O}(n)$).
Without parameter d: $n^{0.8k+o(k)}k^{\mathcal{O}(1)}$.