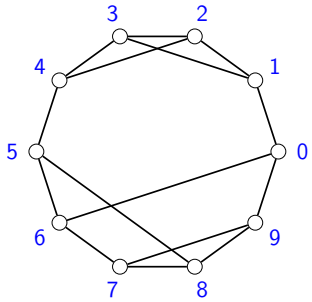
Graph Isomorphism and Beyond

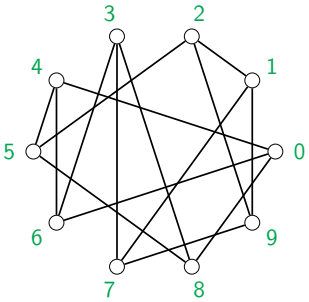
Daniel Neuen

TU Dresden

Workshop "Parameterized Complexity of Computational Reasoning"

Graph Isomorphism Problem

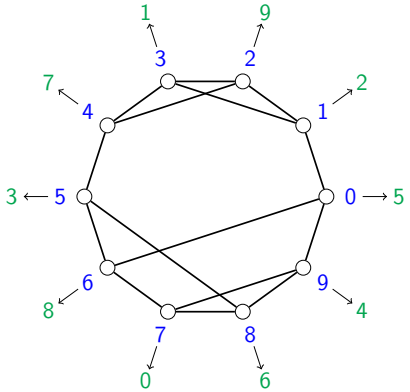


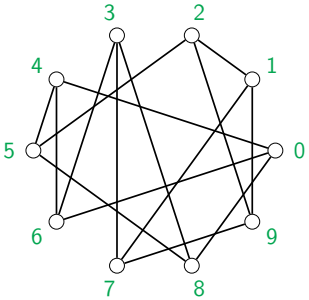


Graph Isomorphism Problem (GI)

Input: Two graphs G_1 and G_2
Problem: Is $G_1 \cong G_2$?

Graph Isomorphism Problem





Graph Isomorphism Problem (GI)

Input: Two graphs G_1 and G_2
Problem: Is $G_1 \cong G_2$?

Some Facts

- GI is in NP
- GI is in coAM
- GI can trivially be solved in time $O(n! \cdot n^2)$

Some Facts

- GI is in NP
- GI is in coAM
- GI can trivially be solved in time $O(n! \cdot n^2)$

Theorem (Babai 2016)

GI can be solved in time $n^{O((\log n)^2)}$.

Canonization

Canonization

A **canonization** is a mapping $\kappa\colon \mathcal{G} \to \mathcal{G}$ such that

- $\kappa(G) \cong G$ for all graphs G, and
- $$\kappa(G) = \kappa(H) \quad \iff \quad G \cong H$$
  for all graphs G, H

Canonization

Canonization

A **canonization** is a mapping $\kappa\colon \mathcal{G} \to \mathcal{G}$ such that

- $\kappa(G) \cong G$ for all graphs G, and
- $$\kappa(G) = \kappa(H) \quad \iff \quad G \cong H$$
  for all graphs G, H

Theorem (Babai 2019)

There exists a canonization that can be computed in time $n^{O((\log n)^c)}$.

Restricted Graph Classes

Many graph classes admit polynomial-time isomorphism tests:

- planar graphs [Hopcroft, Wong '74]
- graphs of maximum degree d [Luks '82]
- graphs of bounded tree-width [Bodlaender '88]
- graphs excluding a fixed minor [Ponomarenko '91]
- graphs excluding a fixed topological subgraph [Grohe, Marx '12]
- graphs of bounded rank-width [Grohe, N. '19]

...

Parameterized Isomorphism Tests

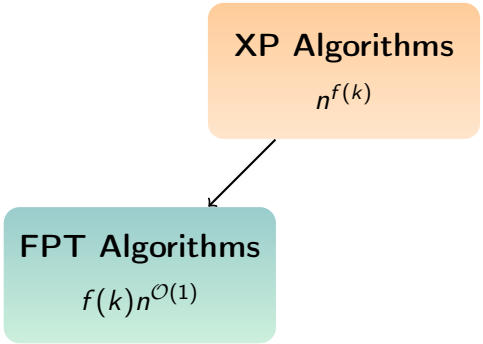
Let k denote a graph parameter (such as maximum degree, genus, tree-width, Hadwiger number, rank-with, etc.)

XP Algorithms

$n^{f(k)}$

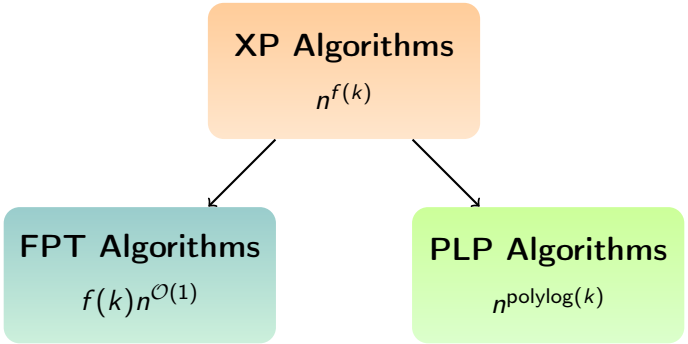
Parameterized Isomorphism Tests

Let k denote a graph parameter (such as maximum degree, genus, tree-width, Hadwiger number, rank-with, etc.)



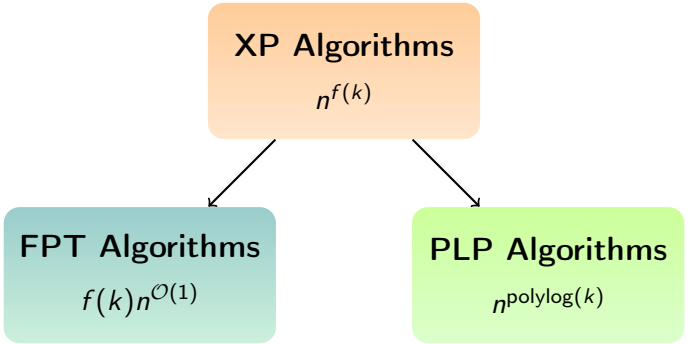
Parameterized Isomorphism Tests

Let k denote a graph parameter (such as maximum degree, genus, tree-width, Hadwiger number, rank-with, etc.)



Parameterized Isomorphism Tests

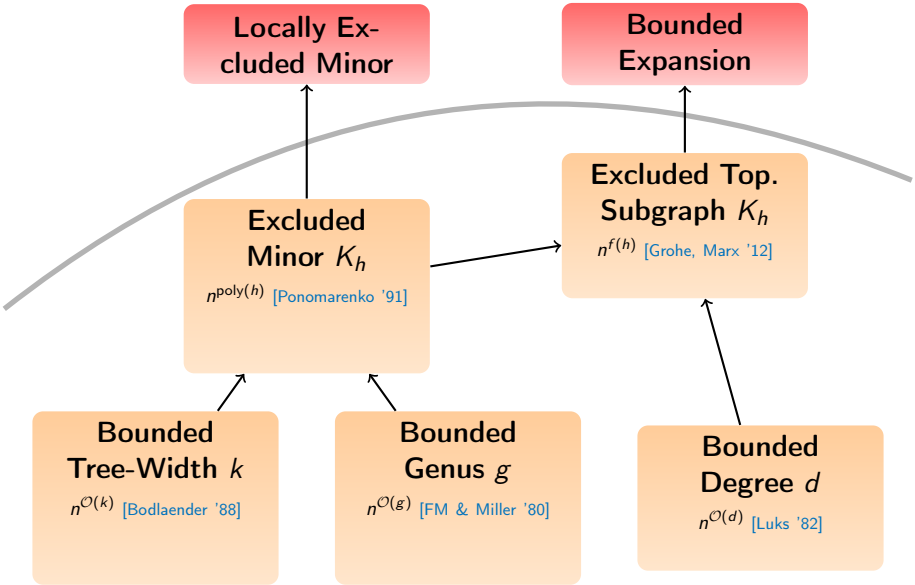
Let k denote a graph parameter (such as maximum degree, genus, tree-width, Hadwiger number, rank-with, etc.)



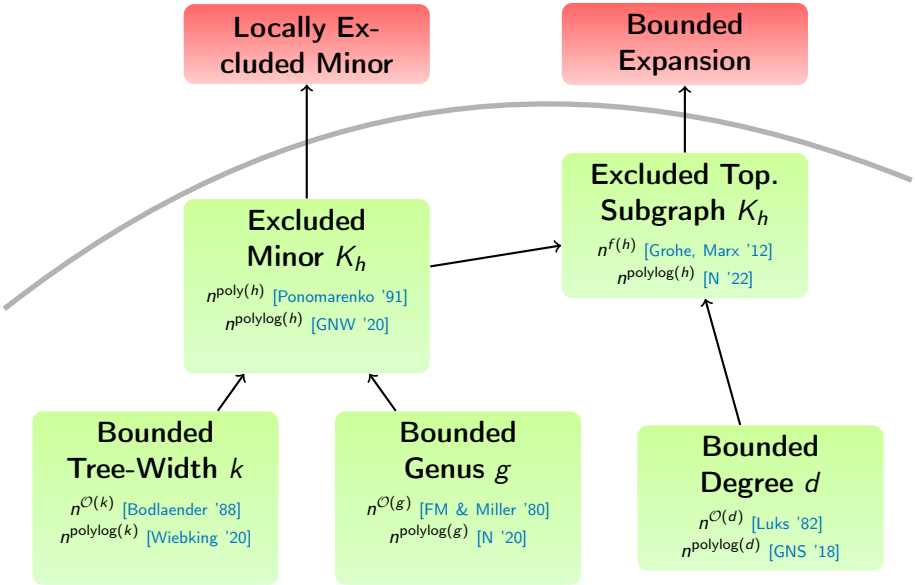
$n^{\text{polylog}(k)} = n^{(\log k)^{\mathcal{O}(1)}}$

$2^k n$ and $n^{\log k}$ are incomparable (consider $k = (\log n)^{1+\varepsilon}$)

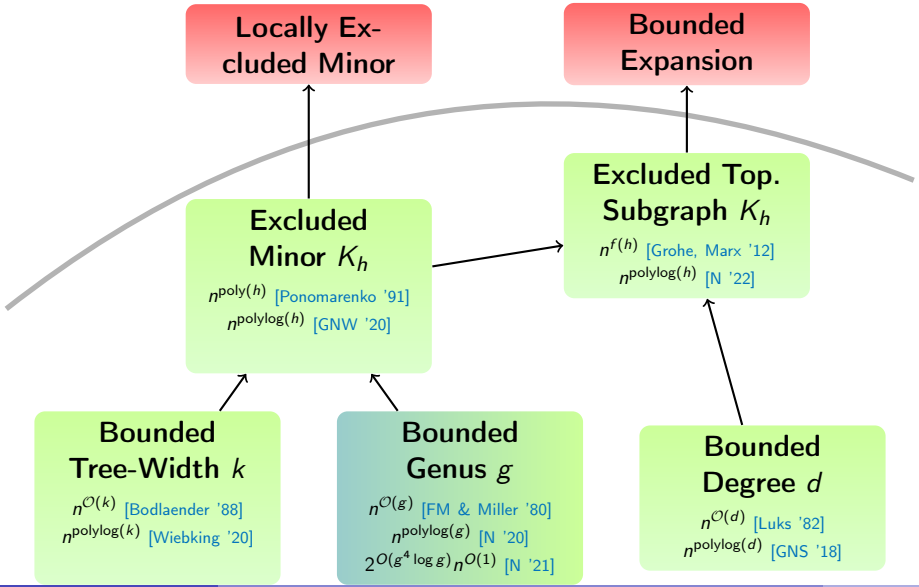
Isomorphism Tests for Sparse Graphs



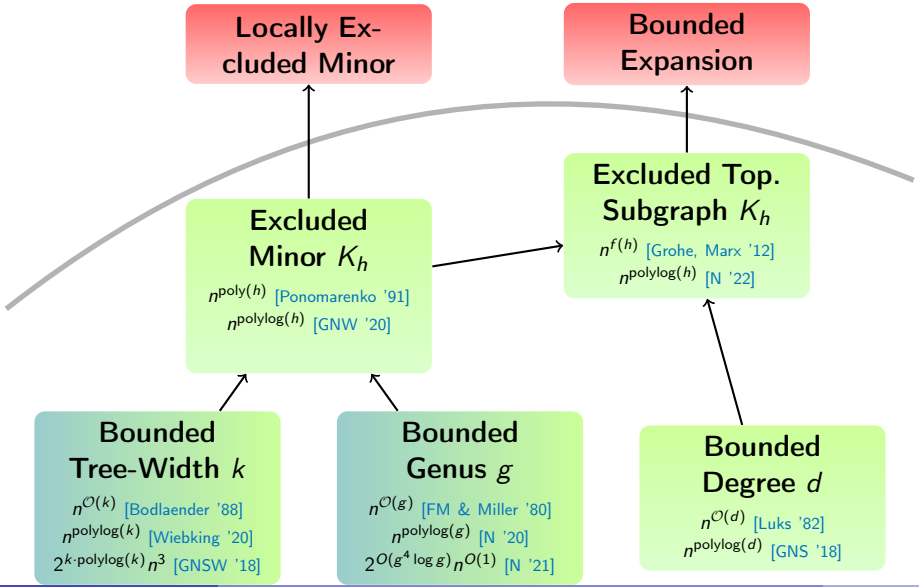
Isomorphism Tests for Sparse Graphs



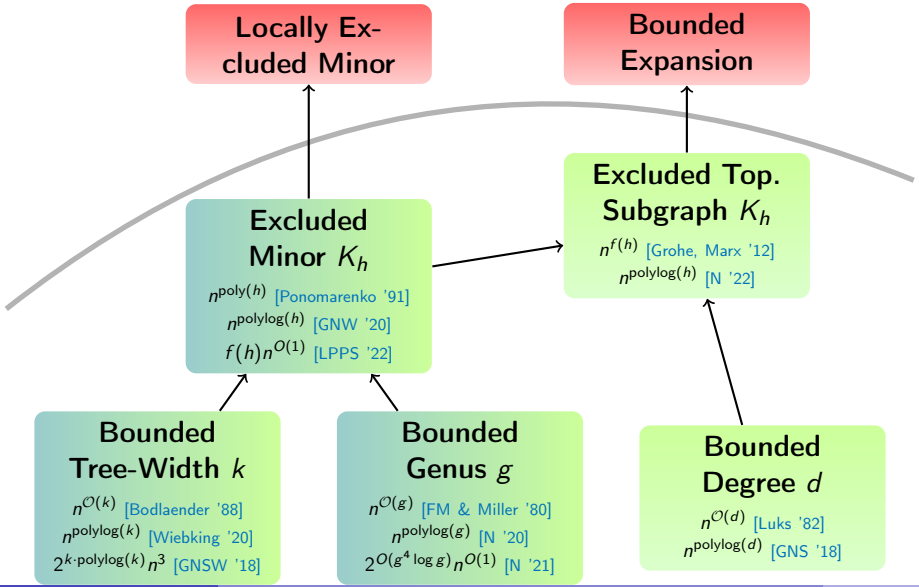
Isomorphism Tests for Sparse Graphs



Isomorphism Tests for Sparse Graphs



Isomorphism Tests for Sparse Graphs



The Isomorphic Priors Model [Dorfman, Döring, Herold, Nanongkai, N., Spoerhase, Wu '26]

Can we solve instances faster if we have them before?

The Isomorphic Priors Model [Dorfman, Döring, Herold, Nanongkai, N., Spoerhase, Wu '26]

Can we solve instances faster if we have them before?

- Consider a computational problem Π on graphs (e.g., Independent Set)

The Isomorphic Priors Model [Dorfman, Döring, Herold, Nanongkai, N., Spoerhase, Wu '26]

Can we solve instances faster if we have them before?

- Consider a computational problem Π on graphs (e.g., Independent Set)
- Suppose that in the past we have already solved instances $G_1, \ldots, G_k$ with optimal solutions $\mathrm{OPT}_1, \ldots, \mathrm{OPT}_k$

The Isomorphic Priors Model [Dorfman, Döring, Herold, Nanongkai, N., Spoerhase, Wu '26]

Can we solve instances faster if we have them before?

- Consider a computational problem Π on graphs (e.g., Independent Set)
- Suppose that in the past we have already solved instances $G_1, \ldots, G_k$ with optimal solutions $\mathrm{OPT}_1, \ldots, \mathrm{OPT}_k$
- Given an instance H, quickly (e.g., in polynomial time)
  - return an optimal solution for H, or
  - correctly conclude that $H \ncong G_i$ for all $i = 1, \ldots, k$ (and use heavy-computation fall-back)

The Isomorphic Priors Model [Dorfman, Döring, Herold, Nanongkai, N., Spoerhase, Wu '26]

Fix a computational problem Π on graphs.

Preprocessing Phase

Input: A database of graph $\mathcal{S} = \{G_1, \ldots, G_k\}$ of **prior graphs**, and optimal solutions $\mathsf{OPT}_1, \ldots, \mathsf{OPT}_k$.

Task: Build a data structure $\mathcal{D}$ to accelerate future queries

The Isomorphic Priors Model [Dorfman, Döring, Herold, Nanongkai, N., Spoerhase, Wu '26]

Fix a computational problem Π on graphs.

Preprocessing Phase

Input: A database of graph $\mathcal{S} = \{G_1, \ldots, G_k\}$ of **prior graphs**, and optimal solutions $\mathrm{OPT}_1, \ldots, \mathrm{OPT}_k$.

Task: Build a data structure $\mathcal{D}$ to accelerate future queries

Query Phase

Input: A graph H.

Task: Given access to $\mathcal{D}$

(a) find an optimal solution for H, or

(b) deduce that $H \ncong G_i$ for all $i = 1, \ldots, k$.

Naive Approaches

Solve Π Directly:

Preprocessing Phase: Do nothing $O(1)$

Query Phase: Solve Π on H $T_\Pi(n, m)$

Naive Approaches

Solve Π Directly:

Preprocessing Phase: Do nothing $O(1)$

Query Phase: Solve Π on H $T_\Pi(n, m)$

Use GI/Canonization Algorithms:

Let $\kappa\colon \mathcal{G} \to \mathcal{G}$ be a canonization that can be computed in time $T_{\text{canon}}(n, m)$

Preprocessing Phase: For each $i \in \{1, \ldots, k\}$ insert $\kappa(G_i)$ (viewed as a string) into a trie with value OPT_i $O(k \cdot T_{\text{canon}}(n, m))$

Query Phase: look up $\kappa(H)$ in the trie $O(T_{\text{canon}}(n, m))$

How to Improve?

Idea: Build **solution-preserving signatures** $\sigma : \mathcal{G} \to \{0,1\}^*$

- if $G \cong H$, then $f(G) = f(H)$
- if $f(G) = f(H)$, then $\mathrm{OPT}(G) = \mathrm{OPT}(H)$

How to Improve?

Idea: Build **solution-preserving signatures** $\sigma \colon \mathcal{G} \to \{0,1\}^*$

- if $G \cong H$, then $f(G) = f(H)$
- if $f(G) = f(H)$, then $\mathsf{OPT}(G) = \mathsf{OPT}(H)$

Preprocessing Phase: For each $i \in \{1, \ldots, k\}$ insert $\sigma(G_i)$ into a trie with value OPT_i $O(k \cdot S_\Pi(n,m))$

Query Phase: look up $\sigma(H)$ in the trie $O(S_\Pi(n,m))$

The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

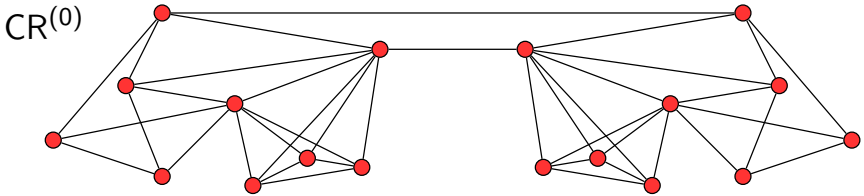
Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

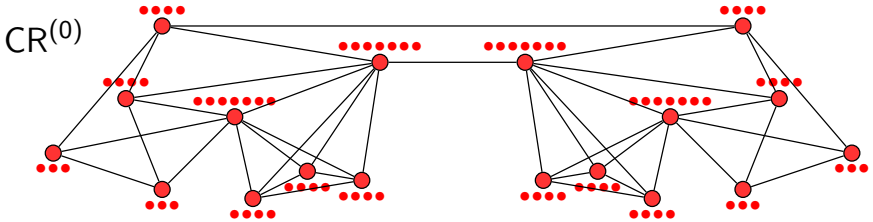


The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

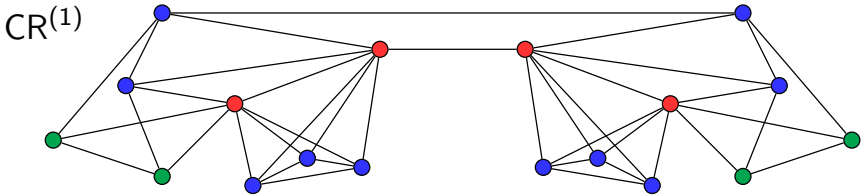


The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

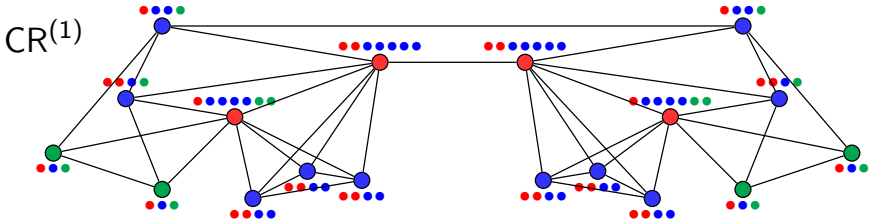


The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

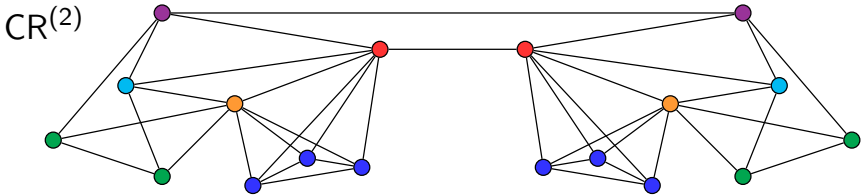


The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

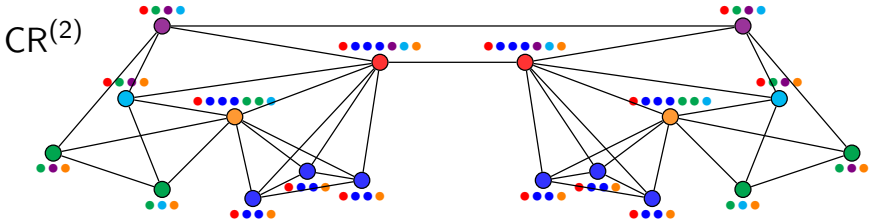


The Color Refinement Algorithm

Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

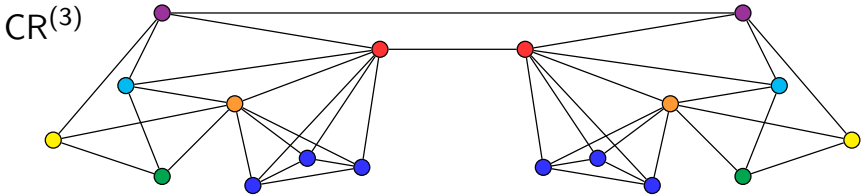


The Color Refinement Algorithm

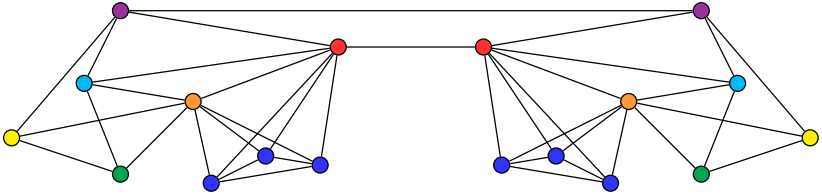
Color Refinement Algorithm (CR)

Initialization Every vertex gets the same color assigned

Refinement Two vertices v, w get different colors assigned if there is some color class C such that $|N(v) \cap C| \neq |N(w) \cap C|$.

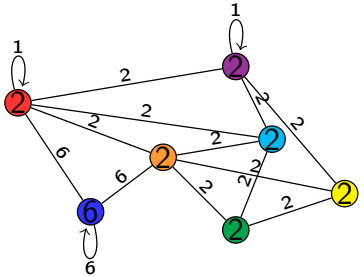


Signatures Based on CR



There is a mapping $\sigma_{\mathrm{CR}} \colon \mathcal{G} \to \{0,1\}^*$ such that

- $G \equiv_{\mathrm{CR}} H \iff \sigma_{\mathrm{CR}}(G) = \sigma_{\mathrm{CR}}(H)$
- $\sigma_{\mathrm{CR}}(G)$ can be computed in time $O((n+m)\log n)$



Constrained Shortest Path

Constrained Shortest Path

Input: A graph $G = (V, E)$, edge lengths $\ell \colon E \to \mathbb{R}_{\geq 0}$,
edge cost $c \colon E \to \mathbb{R}_{\geq 0}$, $s, t \in V$, budget $B \in \mathbb{R}_{\geq 0}$

Task: Find the minimum length of an s-t path of cost at most B?

Constrained Shortest Path

Constrained Shortest Path

Input: A graph $G = (V, E)$, edge lengths $\ell \colon E \to \mathbb{R}_{\geq 0}$, edge cost $c \colon E \to \mathbb{R}_{\geq 0}$, $s, t \in V$, budget $B \in \mathbb{R}_{\geq 0}$

Task: Find the minimum length of an s-t path of cost at most B?

- Constrained Shortest Path is NP-complete
- CR-equivalent instances achieve the same OPT
  - encode edge costs/lengths into edge labels, and s, t into vertex labels

Constrained Shortest Path

Constrained Shortest Path

Input: A graph $G = (V, E)$, edge lengths $\ell\colon E \to \mathbb{R}_{\geq 0}$, edge cost $c\colon E \to \mathbb{R}_{\geq 0}$, $s, t \in V$, budget $B \in \mathbb{R}_{\geq 0}$

Task: Find the minimum length of an s-t path of cost at most B?

- Constrained Shortest Path is NP-complete
- CR-equivalent instances achieve the same OPT
  - encode edge costs/lengths into edge labels, and s, t into vertex labels

Theorem

Constrained Shortest Path can be solved in the Isomorphic Priors Model in $O(k(n+m)\log n)$ preprocessing time and $O((n+m)\log n)$ query time.

Negative Triangle

Negative Triangle

Input: A graph $G = (V, E)$, edge weights $c\colon E \to \mathbb{Z}$
Question: Is there a negative triangle?

No algorithm with running time $O(n^{3-\varepsilon})$ is known.

Negative Triangle

Negative Triangle

Input: A graph $G = (V, E)$, edge weights $c\colon E \to \mathbb{Z}$
Question: Is there a negative triangle?

No algorithm with running time $O(n^{3-\varepsilon})$ is known.

Theorem

Negative Triangle can be solved in the Isomorphic Priors Model in $\widetilde{O}(k \cdot n^{\omega})$ preprocessing time and $\widetilde{O}(n^{\omega})$ randomized query time.

Hard Problems

The following problems cannot be solved in polynomial time in the Isomorphic Priors Model, unless GI can be solved in polynomial time.

- Independent Set
- Clique
- 3-Coloring
- Hamiltonian Cycle
- Feedback Vertex Set
- Max Cut

...

Isomorphic Priors - Discussion

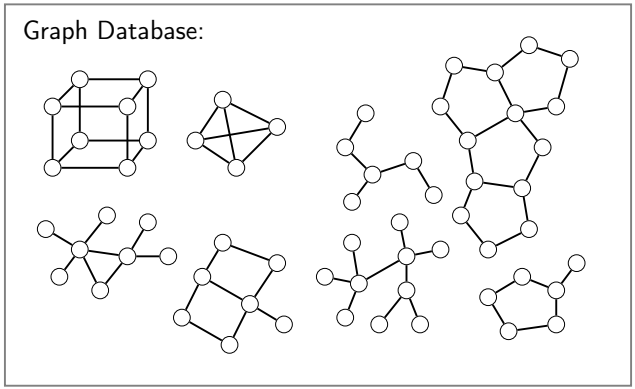
Isomorphic vs Similar Graphs

- Output approximately optimal solution if a prior is close to the query graph

Parameterized Complexity

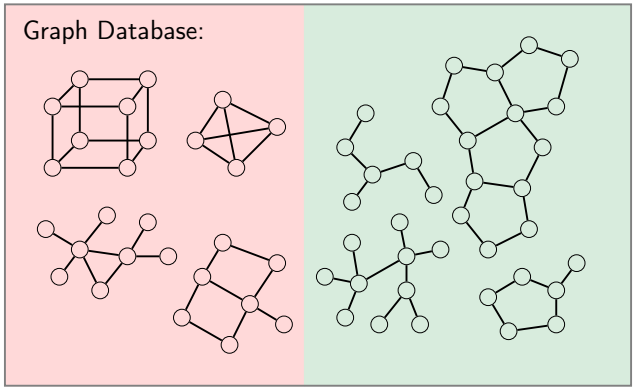
- Is Independent Set solvable in FPT time in the Isomorphic Priors Model?

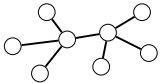
Graph Similarity



Is this graph in the database up to a small error?

Graph Similarity





How to classify this graph?

Edit Distance

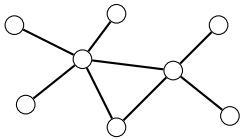
Measure the number of operations to make graphs isomorphic.

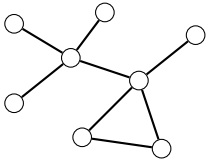
Edit Distance

Measure the number of operations to make graphs isomorphic.

Graph Edit Distance

Minimal number $\delta_{\text{edit}}(G, H)$ of edge deletions and additions to make G, H isomorphic.



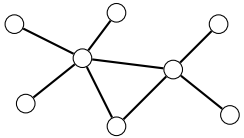


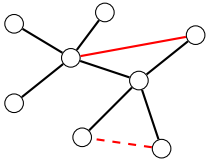
Edit Distance

Measure the number of operations to make graphs isomorphic.

Graph Edit Distance

Minimal number $\delta_{\text{edit}}(G, H)$ of edge deletions and additions to make G, H isomorphic.



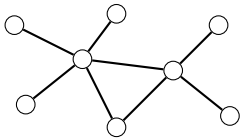


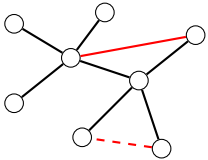
Edit Distance

Measure the number of operations to make graphs isomorphic.

Graph Edit Distance

Minimal number $\delta_{\text{edit}}(G, H)$ of edge deletions and additions to make G, H isomorphic.





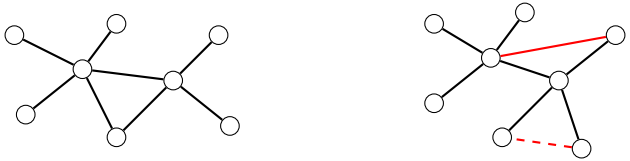
We assume throughout that $|V(G)| = |V(H)|$.

Edit Distance

Measure the number of operations to make graphs isomorphic.

Graph Edit Distance

Minimal number $\delta_{\text{edit}}(G, H)$ of edge deletions and additions to make G, H isomorphic.



We assume throughout that $|V(G)| = |V(H)|$.

Alternative View: Minimize $\|A_G^\pi - A_H\|$ over all permutations π.

How to Measure Graph Similarity

Many Ways to Measure Similarity

- graph edit distance
- minimize $\|A_G^\pi - A_H\|$ for other matrix norms
- distance based on spectrum
- distance based on counting small patterns
- Weisfeiler-Leman kernels

...

How to Measure Graph Similarity

Many Ways to Measure Similarity

- graph edit distance
- minimize $\|A_G^\pi - A_H\|$ for other matrix norms
- distance based on spectrum
- distance based on counting small patterns
- Weisfeiler-Leman kernels

...

Appropriate Measure Depends on Application

How to Measure Graph Similarity

Many Ways to Measure Similarity

- graph edit distance
- minimize $\|A_G^\pi - A_H\|$ for other matrix norms
- distance based on spectrum
- distance based on counting small patterns
- Weisfeiler-Leman kernels

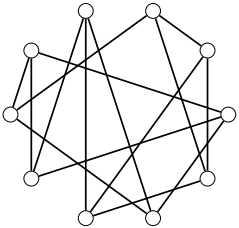
...

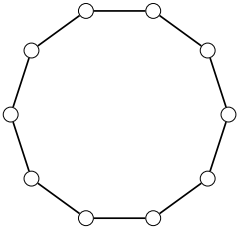
Appropriate Measure Depends on Application

Computational Complexity Can Vary a Lot

Hardness of Graph Edit Distance

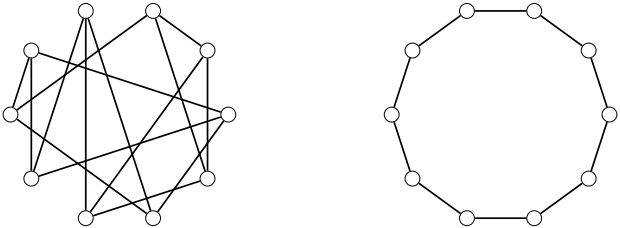
- computing Graph Edit Distance is NP-hard





Hardness of Graph Edit Distance

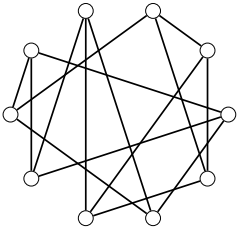
- computing Graph Edit Distance is NP-hard

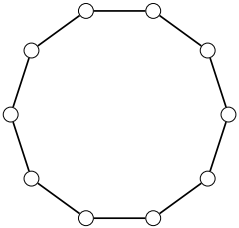


- computing Graph Edit Distance on trees is NP-hard [Grohe, Rattan, Woeginger '18]

Hardness of Graph Edit Distance

- computing Graph Edit Distance is NP-hard





- computing Graph Edit Distance on trees is NP-hard [Grohe, Rattan, Woeginger '18]
- hard to approximate [Arvind, Köbler, Kuhnert, Vasudev '12]

Additive Approximation

Theorem (Arora, Frieze, and Kaplan '02)

There is an algorithm that, given graphs G, H and $\varepsilon > 0$, finds a bijection $\pi\colon V(G) \to V(H)$ such that

$$\|A_G^\pi - A_H\| \leq \delta_{edit}(G, H) + \varepsilon n^2$$

in time $n^{O(\varepsilon^{-2} \log n)}$.

Improvement for Bounded VC Dimension

Let $\mathcal{X}$ be a set system over V.

- $A \subseteq V$ is **shattered** if $\{A \cap X \mid X \in \mathcal{X}\} = 2^A$
- The **VC dimension** of $\mathcal{X}$ is maximum size of a shattered set

The VC dimension of a graph G is the VC dimension of $\{N_G(v) \mid v \in V(G)\}$.

Improvement for Bounded VC Dimension

Let $\mathcal{X}$ be a set system over V.

- $A \subseteq V$ is **shattered** if $\{A \cap X \mid X \in \mathcal{X}\} = 2^A$
- The **VC dimension** of $\mathcal{X}$ is maximum size of a shattered set

The VC dimension of a graph G is the VC dimension of $\{N_G(v) \mid v \in V(G)\}$.

Theorem (Dahan, Grohe, N., Novotny '26)

There is an algorithm that, given graphs G, H of VC dimension d and $\varepsilon > 0$, finds a bijection $\pi \colon V(G) \to V(H)$ such that

$$\|A_G^\pi - A_H\| \leq \delta_{edit}(G, H) + \varepsilon n^2$$

in time $n^{O(\varepsilon^{-2} d)}$.

Robust GI

ε-GI

Input:	Two graphs G and H
Promise:	$G \cong H$ or $\delta_{\text{edit}}(G, H) > \varepsilon n^2$
Question:	Is $G \cong H$?

Robust GI

ε-GI

Input: Two graphs G and H
Promise: $G \cong H$ or $\delta_{\text{edit}}(G, H) > \varepsilon n^2$
Question: Is $G \cong H$?

Theorem (Dahan, Grohe, N., Novotny '26)

ε-GI on graphs of VC dimension at most d can be solved in time $n^{O(\varepsilon^{-1} d \log \varepsilon^{-1})}$.

Idea

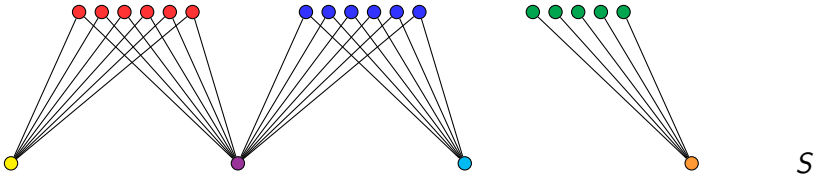
- Consider the set system $\{N_G(v) \triangle N_G(w) \mid v, w \in V(G)\}$; it has VC dimension $O(d)$

Idea

- Consider the set system $\{N_G(v) \triangle N_G(w) \mid v, w \in V(G)\}$; it has VC dimension $O(d)$
- Pick an $\varepsilon/2$-net S of size $O(\varepsilon^{-1} d \log \varepsilon^{-1})$

Idea

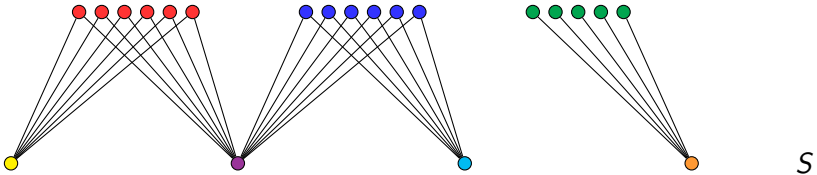
- Consider the set system $\{N_G(v) \triangle N_G(w) \mid v, w \in V(G)\}$; it has VC dimension $O(d)$
- Pick an $\varepsilon/2$-net S of size $O(\varepsilon^{-1} d \log \varepsilon^{-1})$
- For every mapping $\pi\colon S \to V(H)$, run CR after fixing all vertices in S



S

Idea

- Consider the set system $\{N_G(v) \triangle N_G(w) \mid v, w \in V(G)\}$; it has VC dimension $O(d)$
- Pick an $\varepsilon/2$-net S of size $O(\varepsilon^{-1} d \log \varepsilon^{-1})$
- For every mapping $\pi\colon S \to V(H)$, run CR after fixing all vertices in S

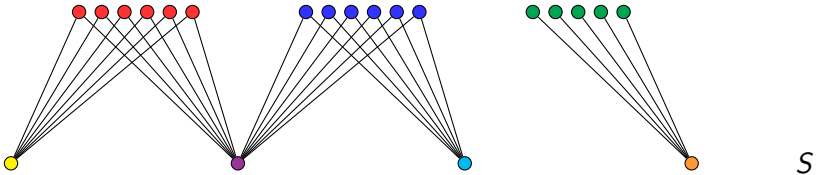


S

- vertices of the same color have similar neighborhoods

Idea

- Consider the set system $\{N_G(v) \triangle N_G(w) \mid v, w \in V(G)\}$; it has VC dimension $O(d)$
- Pick an $\varepsilon/2$-net S of size $O(\varepsilon^{-1} d \log \varepsilon^{-1})$
- For every mapping $\pi \colon S \to V(H)$, run CR after fixing all vertices in S

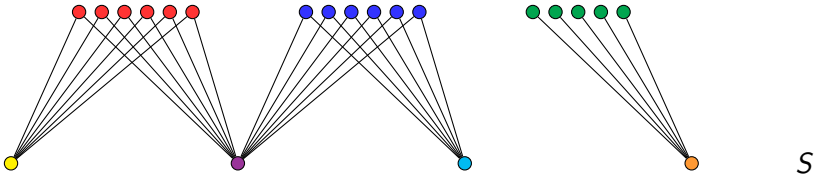


S

- vertices of the same color have similar neighborhoods
- between any pair of color classes, either almost all or almost non of the edges are present

Idea

- Consider the set system $\{N_G(v) \triangle N_G(w) \mid v, w \in V(G)\}$; it has VC dimension $O(d)$
- Pick an $\varepsilon/2$-net S of size $O(\varepsilon^{-1} d \log \varepsilon^{-1})$
- For every mapping $\pi \colon S \to V(H)$, run CR after fixing all vertices in S



S

- vertices of the same color have similar neighborhoods
- between any pair of color classes, either almost all or almost non of the edges are present
- either graphs are distinguished by CR or $\delta_{\text{edit}}(G, H) \leq \varepsilon n^2$

Further Open Problems

Theorem (Dahan, Grohe, N., Novotny '26)

ε-GI on graphs of VC dimension at most d can be solved in time $n^{O(\varepsilon^{-1} d \log \varepsilon^{-1})}$.

Questions:

- Is the dependence on d necessary?
- Can we obtain a running time of $f(\varepsilon, d)n^{O(1)}$?

Parameterized Complexity

What about the parameterized complexity of Graph Edit Distance (parameterized by the edit distance k)?

Parameterized Complexity

What about the parameterized complexity of Graph Edit Distance (parameterized by the edit distance k)?

Proposition (Neuen, Taschin)

Graph Edit Distance is W[1]-hard.

Parameterized Complexity

What about the parameterized complexity of Graph Edit Distance (parameterized by the edit distance k)?

Proposition (Neuen, Taschin)

Graph Edit Distance is W[1]-hard.

What about restricted classes (where graph isomorphism is polynomial-time solvable)?

Question

What is the parameterized complexity of Graph Edit Distance on trees?

Parameterized Complexity

What about the parameterized complexity of Graph Edit Distance (parameterized by the edit distance k)?

Proposition (Neuen, Taschin)

Graph Edit Distance is W[1]-hard.

What about restricted classes (where graph isomorphism is polynomial-time solvable)?

Question

What is the parameterized complexity of Graph Edit Distance on trees?

- What about Graph Cleaning (only edge deletions in G) on graphs of tree-width 2? The problem is FPT on trees [Marx, Schlotter '08].