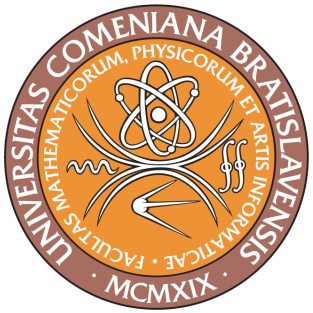
Interplay Between Global and Local Symmetries Computational Aspects

Tatiana Jajcayová

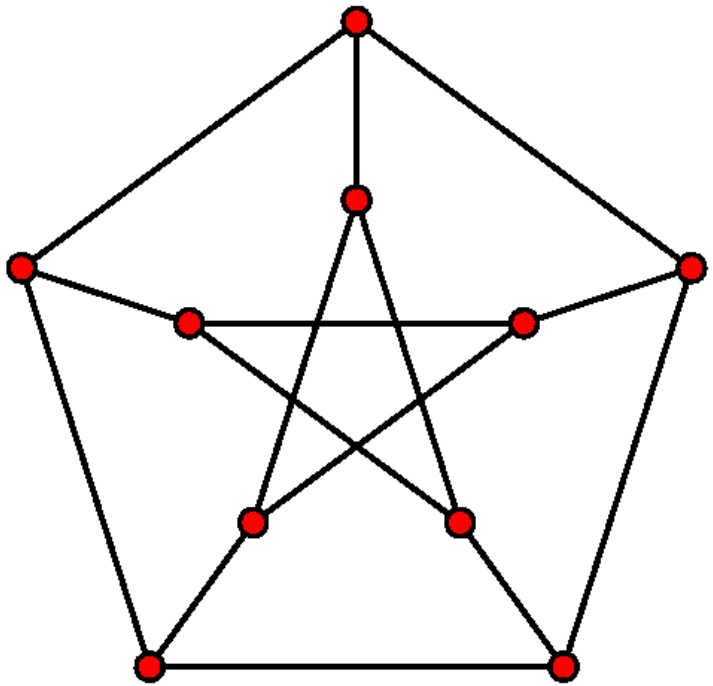


Comenius University, Bratislava

CSD 11
Kranjska Gora, Slovenia
May 4, 2026

Automorphisms vs. Symmetries

An automorphism is an isomorphism from a graph to itself.



The automorphism group captures symmetries of the graph.

Motivation: Symmetries of combinatorial structures

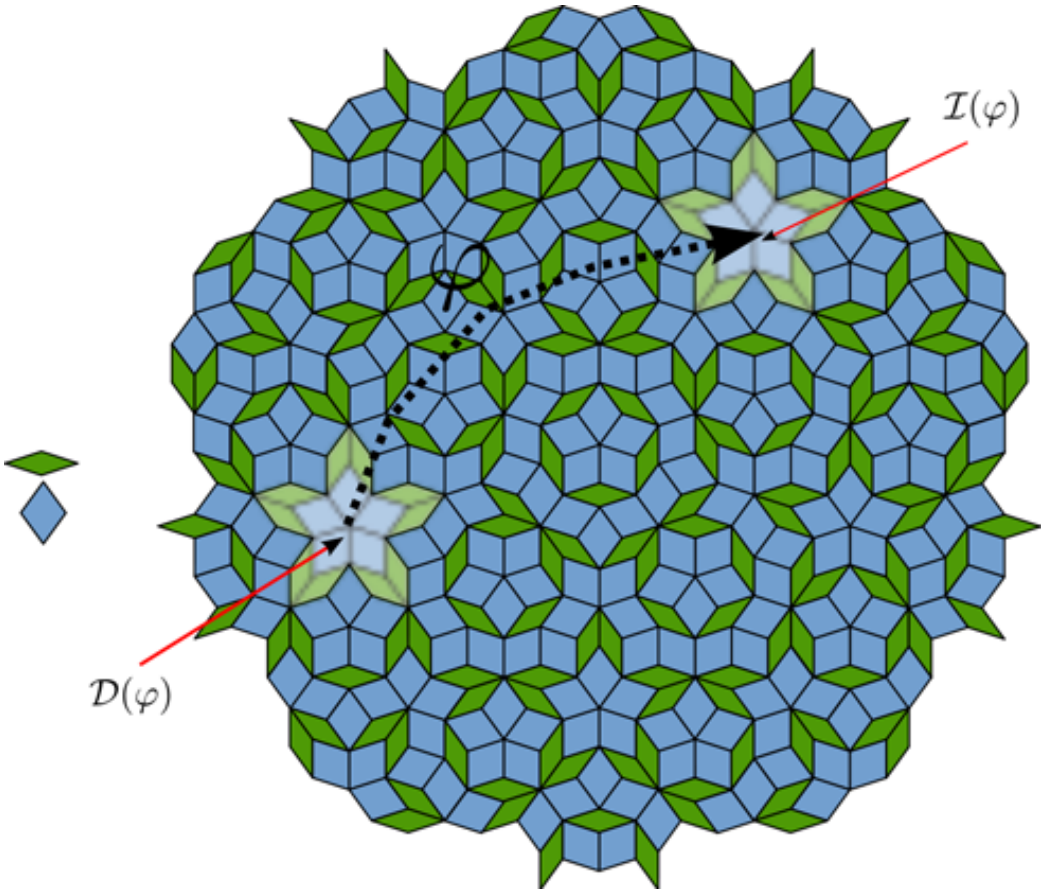
The following problems are polynomially equivalent:

- GI: the graph isomorphism problem
- col-GI: isomorphism problem of colored graphs
- ISO: isomorphism of general combinatorial objects
- Aut(Γ): compute generating set for automorphism group
- $|\mathrm{Aut}(\Gamma)|$: determine the size of Aut(Γ)

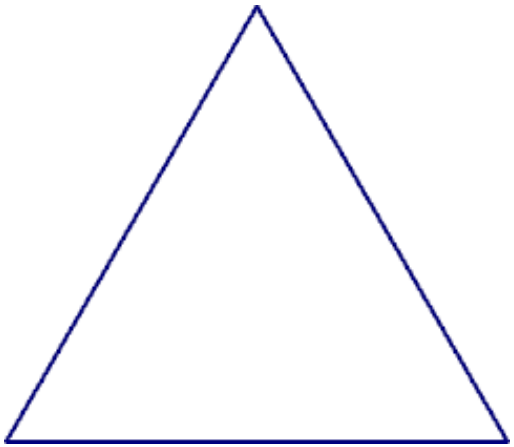
The graph isomorphism problem is the problem of detecting symmetries of combinatorial structures.

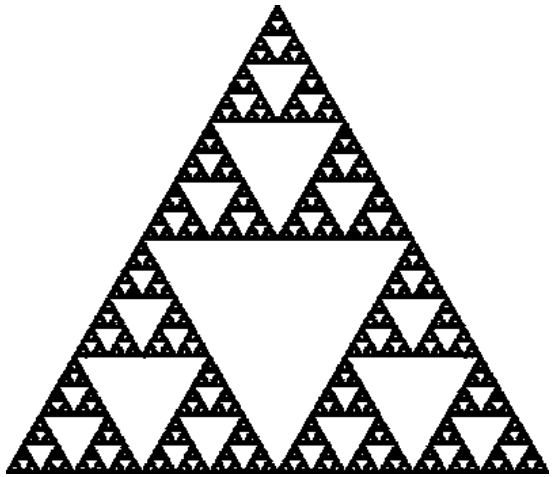
Partial Symmetries

Penrose tiling



Partial Symmetries

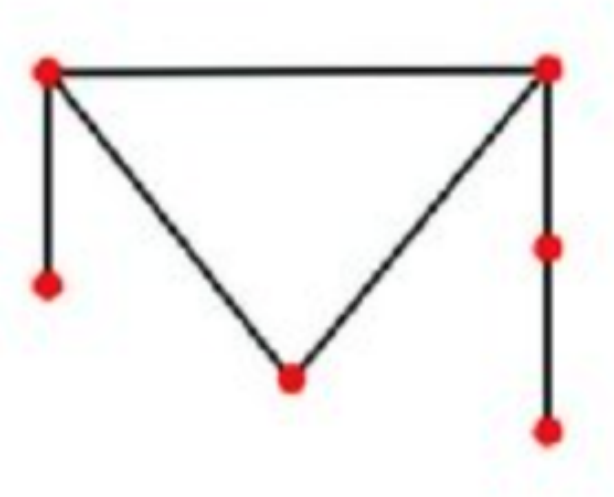




Absence of symmetry

A graph Γ is called *asymmetric* (or rigid) if it does not have non-trivial automorphisms. (i.e. $|Aut(\Gamma)| = 1$).

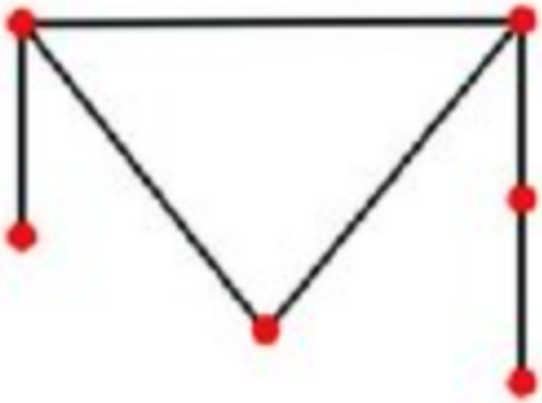
Example:

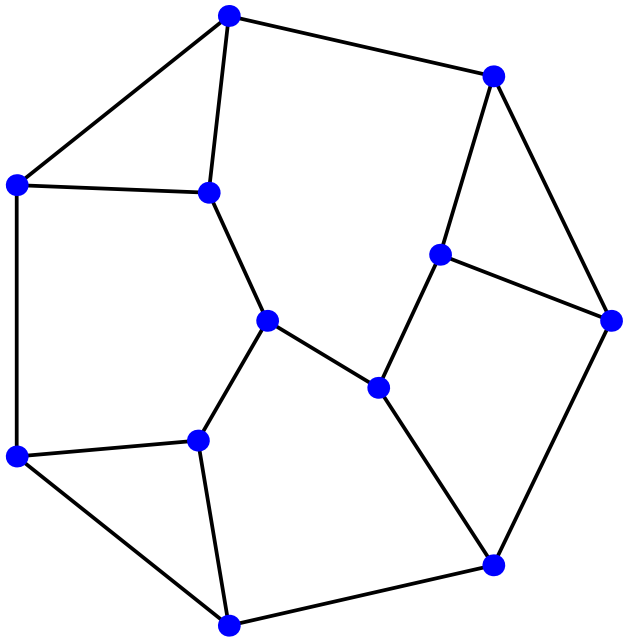


Absence of symmetry

A graph Γ is called *asymmetric* (or rigid) if it does not have non-trivial automorphisms. (i.e. $|Aut(\Gamma)| = 1$).

Example:





Asymmetric graphs

- (Erdös, Rényi, 1963)
  Almost all graphs are asymmetric

Asymmetric graphs

- (Erdös, Rényi, 1963)
  Almost all graphs are asymmetric
- (Kim, Sudakov, Vu, 2002)
  Almost all regular graphs are asymmetric

Asymmetric graphs

- (Erdös, Rényi, 1963)
  Almost all graphs are asymmetric
- (Kim, Sudakov, Vu, 2002)
  Almost all regular graphs are asymmetric

- in particular, $Aut(\Gamma)$ is trivial for all those graphs

Worst case instances for GI testers

[Neuen, Schweitzer, 2017] Algorithms for testing graph isomorphism can in the worst case have exponential running time.

Worst case instances for GI testers

[Neuen, Schweitzer, 2017] Algorithms for testing graph isomorphism can in the worst case have exponential running time.

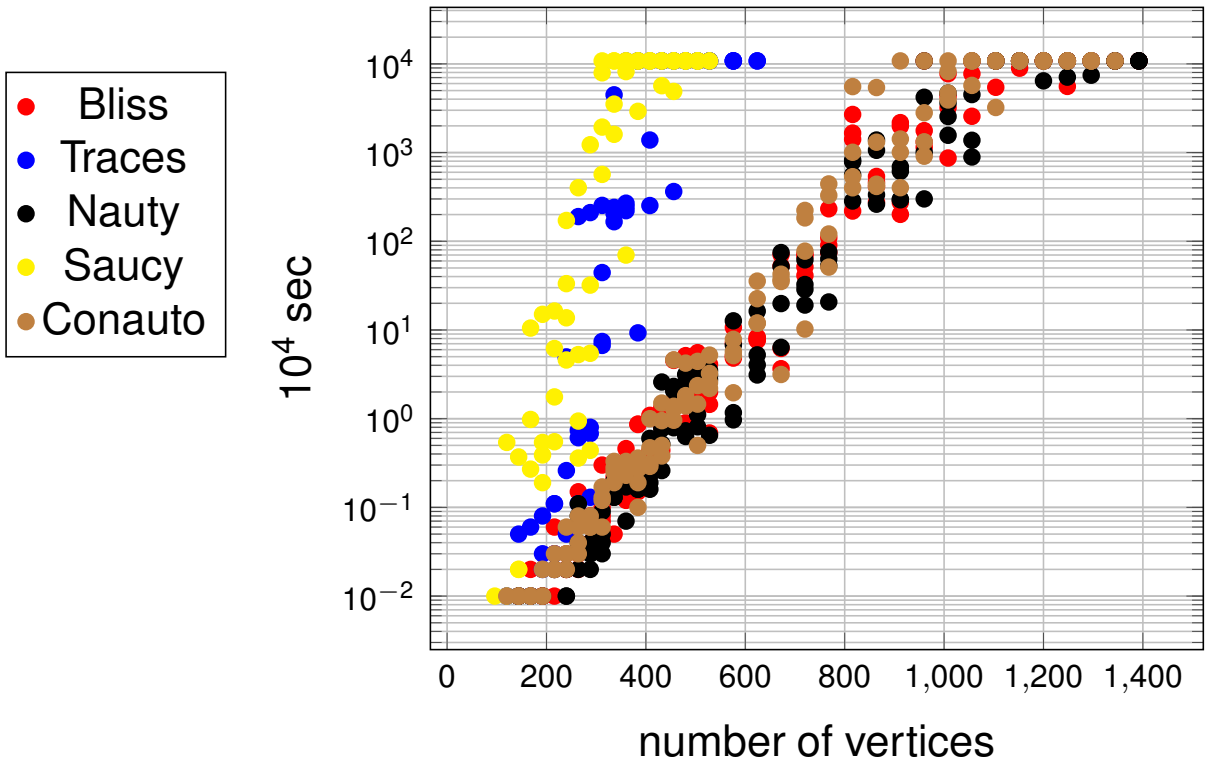


Figure: From (Schweitzer, 2017)

These benchmarks are asymmetric (rigid) graphs.

Asymmetric graphs

Erdös, Rényi:

- Symmetrization: removing (r) and adding (s) edges to make a graph symmetric
- Degree of asymmetry $A(\Gamma)$: the minimum of $r+s$ taken over all possible symmetrizations
- The asymmetry of a graph of order n can not exceed $\frac{n-1}{2}$; and this estimate is asymptotically best possible
- The relative asymmetry of Γ,

$$a(\Gamma) = \frac{A(\Gamma)}{\frac{n-1}{2}}$$

$0 \leq a(\Gamma) \leq 1$

Symmetry vs. Assymetry: Minimal asymmetric graphs

An undirected graph G on at least two vertices is *minimal asymmetric* if G is asymmetric and no proper induced subgraph of G on at least two vertices is asymmetric.

Minimal asymmetric graphs

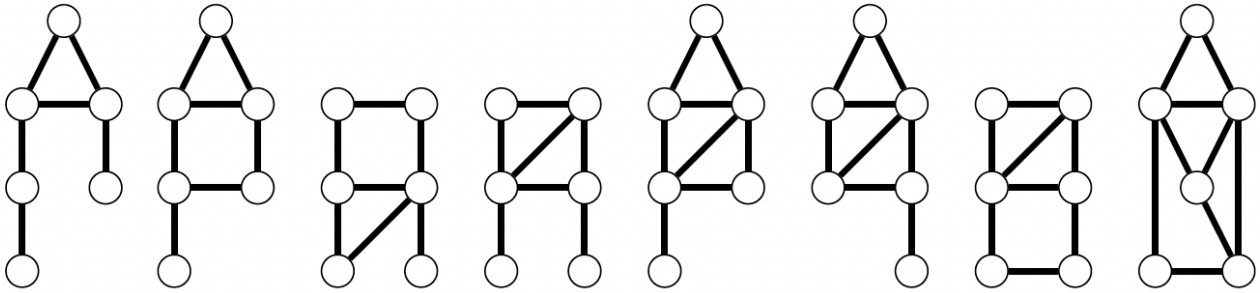
Theorem (Schweitzer, Pascal; Schweitzer, Patrick, 2017)

There are exactly 18 finite minimal asymmetric undirected graphs up to isomorphism.

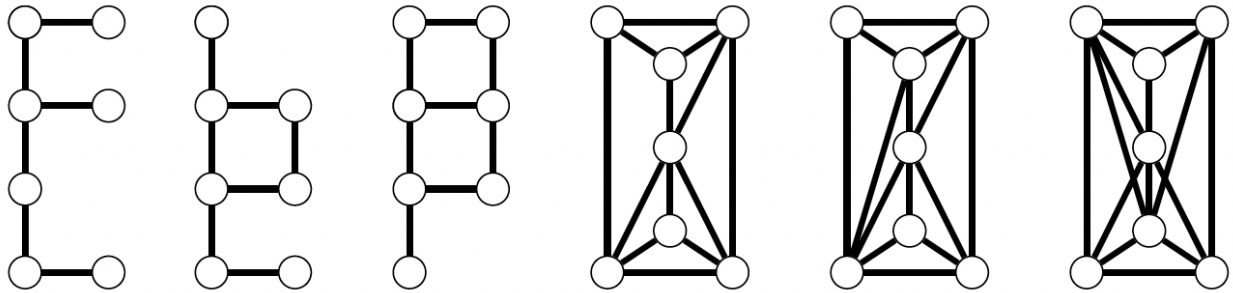
Nešetřil's conjecture: There are exactly 18 minimal asymmetric graphs (coming in 9 complementary pairs).

Nešetřil and Sabidussi earlier established a close connection between minimal asymmetric graphs and minimal involution-free graphs.

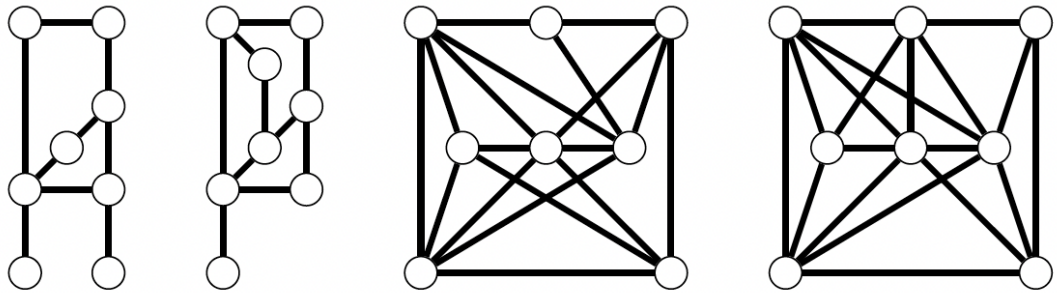
Minimal asymmetric graphs



X_1 $(6, 6, X_8)$ X_2 $(6, 7, X_7)$ X_3 $(6, 7, X_6)$ X_4 $(6, 7, X_5)$ X_5 $(6, 8, X_4)$ X_6 $(6, 8, X_3)$ X_7 $(6, 8, X_2)$ X_8 $(6, 9, X_1)$



X_9 $(7, 6, X_{14})$ X_{10} $(7, 7, X_{13})$ X_{11} $(7, 8, X_{12})$ X_{12} $(7, 13, X_{11})$ X_{13} $(7, 14, X_{10})$ X_{14} $(7, 15, X_9)$



X_{15} $(8, 9, X_{18})$ X_{16} $(8, 10, X_{17})$ X_{17} $(8, 18, X_{16})$ X_{18} $(8, 19, X_{15})$

Symmetry vs. Asymmetry

Cayley graph $\mathcal{C}(G, X)$.

- Cayley graphs are well-known highly symmetric (vertex-transitive) graphs; $G_L \leq Aut(C(G, X))$

Symmetry vs. Asymmetry

Cayley graph $\mathcal{C}(G, X)$.

- Cayley graphs are well-known highly symmetric (vertex-transitive) graphs; $G_L \leq Aut(C(G, X))$
- In case when $G_L = Aut(C(G, X))$, we call the graph a **G**raphical **R**egular **R**epresentation; GRR, $Stab_{Aut(C(G,X))}(u) = \langle 1 \rangle$,

Symmetry vs. Asymmetry

Cayley graph $\mathcal{C}(G, X)$.

- Cayley graphs are well-known highly symmetric (vertex-transitive) graphs; $G_L \leq Aut(C(G, X))$
- In case when $G_L = Aut(C(G, X))$, we call the graph a **G**raphical **R**egular **R**epresentation; GRR, $Stab_{Aut(C(G,X))}(u) = \langle 1 \rangle$,

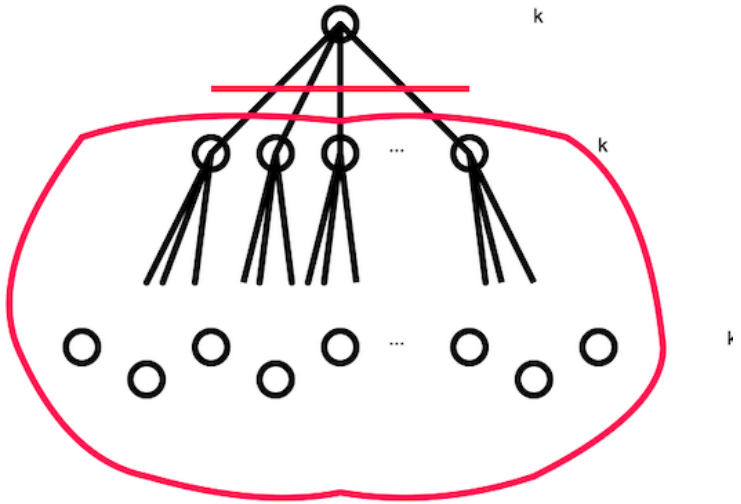
- Removing (any) single vertex (together with its incident edges) from a GRR $C(G, X)$ yields an asymmetric graph with a trivial automorphism group:

Symmetry vs. Asymmetry

Cayley graph $\mathcal{C}(G, X)$.

- Cayley graphs are well-known highly symmetric (vertex-transitive) graphs; $G_L \leq Aut(C(G, X))$
- In case when $G_L = Aut(C(G, X))$, we call the graph a **G**raphical **R**egular **R**epresentation; GRR, $Stab_{Aut(C(G,X))}(u) = \langle 1 \rangle$,

- Removing (any) single vertex (together with its incident edges) from a GRR $C(G, X)$ yields an asymmetric graph with a trivial automorphism group:

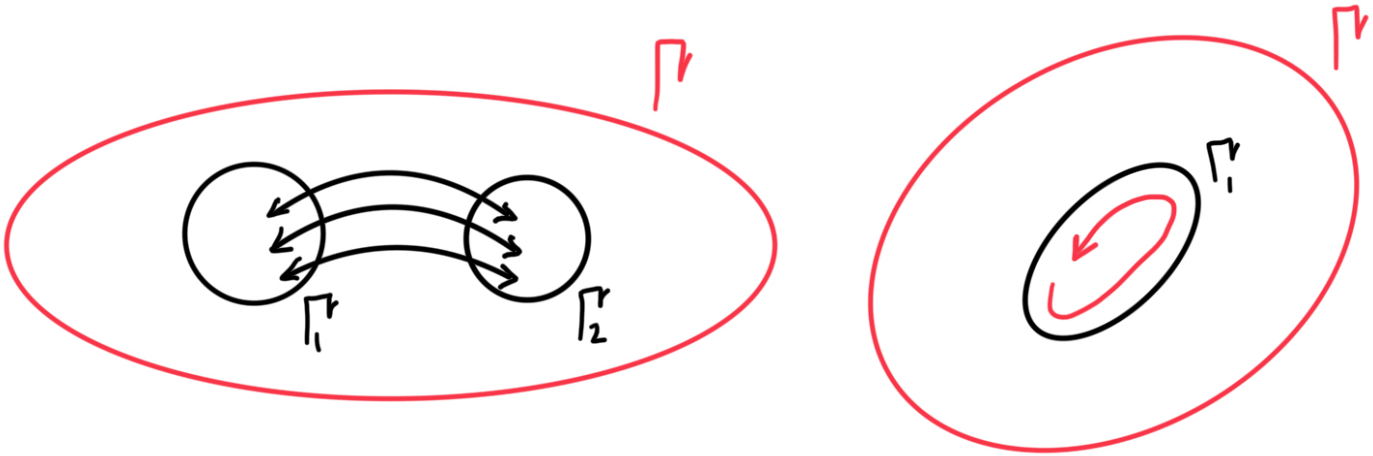


Partial Automorphisms

A partial automorphism of $\Gamma = (V, \mathcal{E})$ is an isomorphism between two *induced* subgraphs.

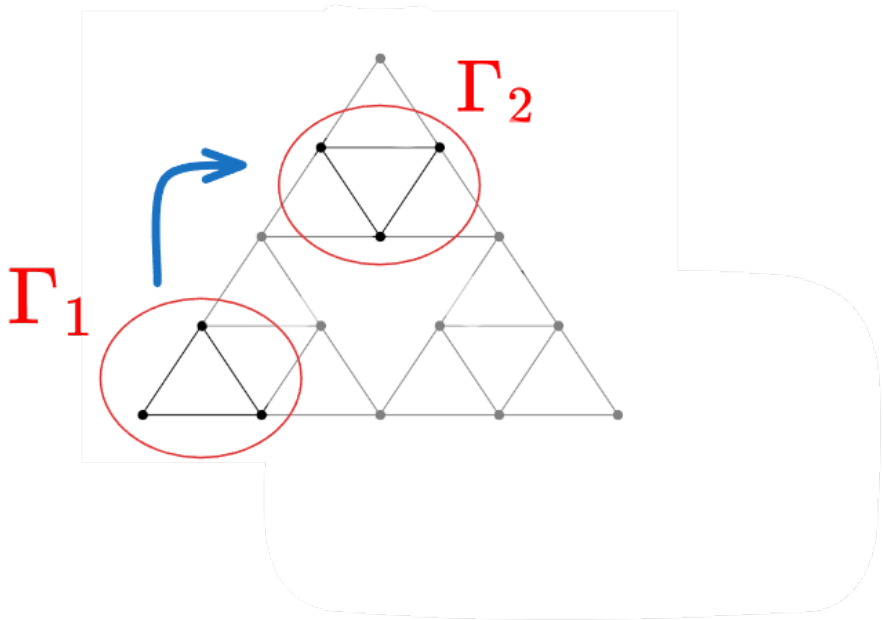
Partial Automorphisms

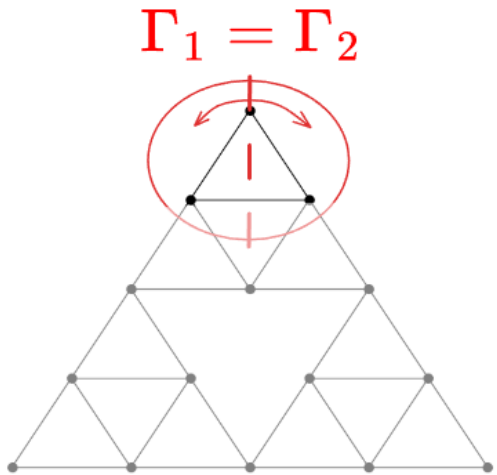
A partial automorphism of $\Gamma = (V, \mathcal{E})$ is an isomorphism between two *induced* subgraphs.



Partial Automorphisms

A partial automorphism of $\Gamma = (V, \mathcal{E})$ is an isomorphism between two *induced* subgraphs.

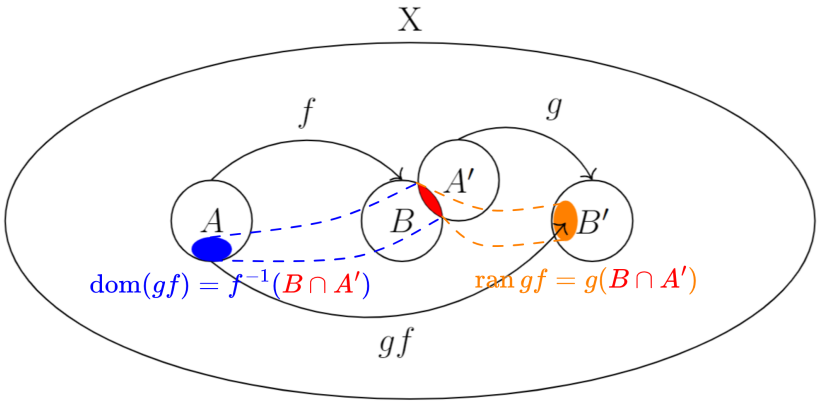




The rank of a partial automorphism is given by the size of its domain.

Inverse Monoid of Partial Automorphisms

The set of all partial automorphisms, denoted PAut(Γ) with the composition and partial inverse of partial maps forms an inverse monoid.



$\mathrm{PAut}(\Gamma) \leq \mathrm{PSym}(V)$

Inverse Monoid

A set together with an associative binary operation is called a *semigroup*

A semigroup having an identity element is a *monoid*

Inverse Monoid

A set together with an associative binary operation is called a *semigroup*
A semigroup having an identity element is a *monoid*

A monoid M is called **inverse**

- if for every $a \in M$ there exists a unique element a^{-1} s.t.

$$a \cdot a^{-1} \cdot a = a$$

$$a^{-1} \cdot a \cdot a^{-1} = a^{-1}$$

"Archetypal" inverse semigroup PSym(X)

PSym(X) - set of all partial permutations of X
= bijections between subsets of X (including $\emptyset$).

$$\varphi : Y \to Z \qquad Y, Z \subseteq X$$

Y - domain $dom\varphi$
Z - range $ran\varphi$

$|dom\varphi| = |ran\varphi|$ - rank of φ

"Archetypal" inverse semigroup PSym(X)

PSym(X) - set of all partial permutations of X
= bijections between subsets of X (including $\emptyset$).

$$\varphi : Y \to Z \qquad Y, Z \subseteq X$$

Y - domain $dom\varphi$
Z - range $ran\varphi$

$|dom\varphi| = |ran\varphi|$ - rank of φ

The cycle notation of classical permutations generalizes by the addition of a notion called a path, which (unlike a cycle) ends when it reaches the "undefined" element.

$$\mathrm{dom}(x_1, x_2 ... x_k] = \{x_1, x_2, ..., x_{k-1}\}$$

$$\mathrm{ran}(x_1, x_2 ... x_k] = \{x_2, x_3, ..., x_k\}$$

PSym(X)

- Operation on PSym(X) - is composition of partial maps:
  Let α and β be partial permutations of a set X; α and β can be composed (from left to right) on the largest domain upon which it "makes sense" to compose them (may be the $\emptyset$)

$$\operatorname{dom} \alpha\beta = [\operatorname{im} \alpha \cap \operatorname{dom} \beta]\alpha^{-1}$$

PSym(X)

- Operation on PSym(X) - is composition of partial maps: Let α and β be partial permutations of a set X; α and β can be composed (from left to right) on the largest domain upon which it "makes sense" to compose them (may be the $\emptyset$)

$$\mathrm{dom}\,\alpha\beta = [\mathrm{im}\,\alpha \cap \mathrm{dom}\,\beta]\alpha^{-1}$$

- Inverse of φ - just the usual inverse φ^{-1} of the bijection $\varphi : dom\varphi \to ran\varphi$

PSym(X)

- Operation on PSym(X) - is composition of partial maps:
  Let α and β be partial permutations of a set X; α and β can be composed (from left to right) on the largest domain upon which it "makes sense" to compose them (may be the $\emptyset$)

$$\mathrm{dom}\,\alpha\beta = [\mathrm{im}\,\alpha \cap \mathrm{dom}\,\beta]\alpha^{-1}$$

- Inverse of φ - just the usual inverse φ^{-1} of the bijection $\varphi : dom\varphi \to ran\varphi$
- Identity of PSym(X) - is id_X

PSym(X)

- Operation on PSym(X) - is composition of partial maps: Let α and β be partial permutations of a set X; α and β can be composed (from left to right) on the largest domain upon which it "makes sense" to compose them (may be the $\emptyset$)

$$\mathrm{dom}\,\alpha\beta = [\mathrm{im}\,\alpha \cap \mathrm{dom}\,\beta]\alpha^{-1}$$

- Inverse of φ - just the usual inverse φ^{-1} of the bijection $\varphi : dom\varphi \to ran\varphi$
- Identity of PSym(X) - is id_X
- Local Identities: $A \subset X$ - id_A, idempotents

PSym(X)

- Operation on PSym(X) - is composition of partial maps: Let α and β be partial permutations of a set X; α and β can be composed (from left to right) on the largest domain upon which it "makes sense" to compose them (may be the $\emptyset$)

$$\mathrm{dom}\,\alpha\beta = [\mathrm{im}\,\alpha \cap \mathrm{dom}\,\beta]\alpha^{-1}$$

- Inverse of φ - just the usual inverse φ^{-1} of the bijection $\varphi : dom\varphi \to ran\varphi$
- Identity of PSym(X) - is id_X
- Local Identities: $A \subset X$ - id_A, idempotents
- Zero - PSym(X) has also zero element - empty map - $id_\emptyset$

Wagner-Preston representation

While groups can be represented as symmetries:

Theorem (Cayley)

Every (finite) group can be represented as a group of permutations of a (finite) set.

Inverse semigroups can be represented as partial symmetries:

Theorem (Wagner-Preston)

Every (finite) inverse semigroup can be represented as the inverse semigroup of partial bijections of a (finite) set.

Restrictions vs. extensions

It is clear that all restrictions of an automorphism of a graph are partial automorphisms.

But not all partial automorphisms extend to a (global) automorphism.

Homogeneous graphs

A graph Γ is *homogeneous* if any isomorphism between induced subgraphs extends to an automorphism of Γ.

Homogeneous graphs

A graph Γ is *homogeneous* if any isomorphism between induced subgraphs extends to an automorphism of Γ.

Theorem (Gardiner 1976)

A finite homogeneous graph is one of the following:

- *a disjoint union of complete graphs of the same size*
- *a regular complete multipartite graph*
- *the 5-cycle*
- *the line graph of $K_{3,3}$*

Extensions of partial automorphisms

Jaroslav Nešetřil, Matěj Konečný, ...

Extension property for partial automorphisms, EPPA

Extensions of partial automorphisms

Jaroslav Nešetřil, Matěj Konečný, ...
Extension property for partial automorphisms, EPPA

Let A be a structure and let B be its (induced) substructure. A is an EPPA-witness for B if every partial automorphism of B extends to an automorphism of A.

A class $\mathcal{C}$ of **finite** structures has EPPA if for every $B \in \mathcal{C}$ there is $A \in \mathcal{C}$, which is an EPPA-witness for B.

Extensions of partial automorphisms

Jaroslav Nešetřil, Matěj Konečný, ...
Extension property for partial automorphisms, EPPA

Let A be a structure and let B be its (induced) substructure. A is an EPPA-witness for B if every partial automorphism of B extends to an automorphism of A.

A class $\mathcal{C}$ of **finite** structures has EPPA if for every $B \in \mathcal{C}$ there is $A \in \mathcal{C}$, which is an EPPA-witness for B.

Theorem (Hrushovski 1992)

The class of all finite graphs has EPPA.

Inverse Monoid of Partial Automorphisms of Graphs

- No finite graph of order greater than 1 has a trivial inverse monoid of partial automorphisms PAut(Γ)

Inverse Monoid of Partial Automorphisms of Graphs

- No finite graph of order greater than 1 has a trivial inverse monoid of partial automorphisms PAut(Γ)
- The inverse monoid PAut(Γ) is a complete algebraic description of Γ

Inverse Monoid of Partial Automorphisms of Graphs

- No finite graph of order greater than 1 has a trivial inverse monoid of partial automorphisms PAut(Γ)
- The inverse monoid PAut(Γ) is a complete algebraic description of Γ
- $PAut(\Gamma)$ contains $Aut(\Gamma)$ as its subgroup
  (the automorphism group of Γ of **order** n consists of the partial automorphisms of **rank** n)

Inverse Monoid of Partial Automorphisms of Graphs

- No finite graph of order greater than 1 has a trivial inverse monoid of partial automorphisms PAut(Γ)
- The inverse monoid PAut(Γ) is a complete algebraic description of Γ
- $PAut(\Gamma)$ contains $Aut(\Gamma)$ as its subgroup (the automorphism group of Γ of **order** n consists of the partial automorphisms of **rank** n)

- Jajcay, Jajcayová, Szakács, and Szendrei (2021) gave a characterization of partial automorphism inverse monoids for graphs.

Structure of inverse monoids

Green's relations:

$s, t \in S$. We define $\mathcal{L}$ and $\mathcal{R}$:

- $s \mathcal{L} t \Leftrightarrow \exists\, x, y \in S$ s.t. $xs = t$ & $yt = s$,
- $s \mathcal{R} t \Leftrightarrow \exists\, x, y \in S$ s.t. $sx = t$ & $ty = s$.

In $PSym(X)$,

$$\varphi_1 \mathcal{L} \varphi_2 \Leftrightarrow \operatorname{dom} \varphi_1 = \operatorname{dom} \varphi_2,$$
$$\varphi_1 \mathcal{R} \varphi_2 \Leftrightarrow \operatorname{ran} \varphi_1 = \operatorname{ran} \varphi_2.$$

Structure of PAut(Γ) for graph Γ

- finding the whole monoid is hard (both in time and memory)

Proposition (R.Jajcay, T.J., N. Szakács, M. Szendrei 2021)

For any graph Γ, *the* $\mathcal{D}$*-classes of* PAut(Γ) *correspond to the isomorphism classes of induced subgraphs of* Γ, *that is, two elements are* $\mathcal{D}$*-related if and only if the subgraphs induced by their respective domains (or images) are isomorphic.*

Partial order for $\mathcal{D}$-classes: "subgraph" relation

Example

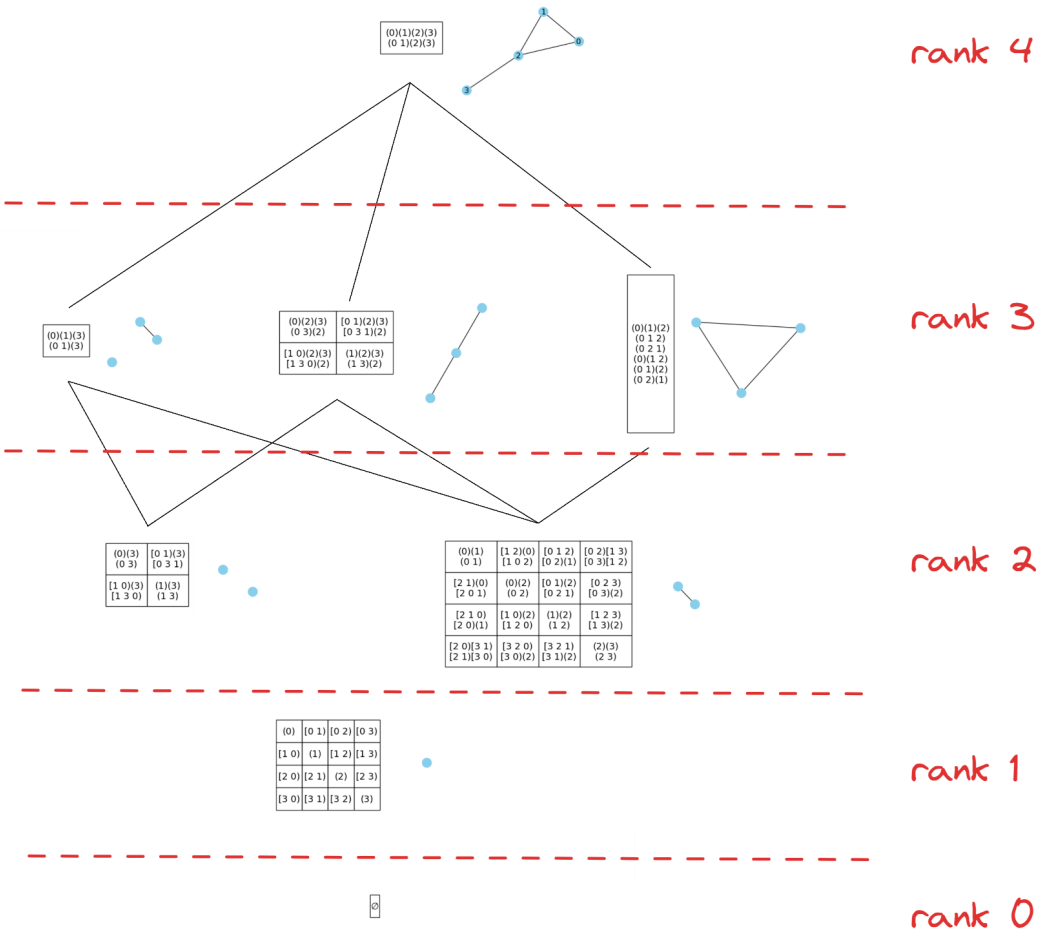


Figure: The Green-class structure of partial graph automorphisms

Asymmetric depth & Symmetry Level of a Graph

Definition

The **asymmetric depth** of a graph Γ of order n is $d(\Gamma) = n - k_{max}$

Asymmetric depth & Symmetry Level of a Graph

Definition

The **asymmetric depth** of a graph Γ of order n is $d(\Gamma) = n - k_{max}$

Definition

The **level of symmetry** of a graph Γ of order n is the ratio between the largest rank of a non-trivial partial automorphism of Γ and its order n.

- A graph Γ admitting a non-trivial automorphism has level of symmetry 1

Asymmetric depth & Symmetry Level of a Graph

Definition

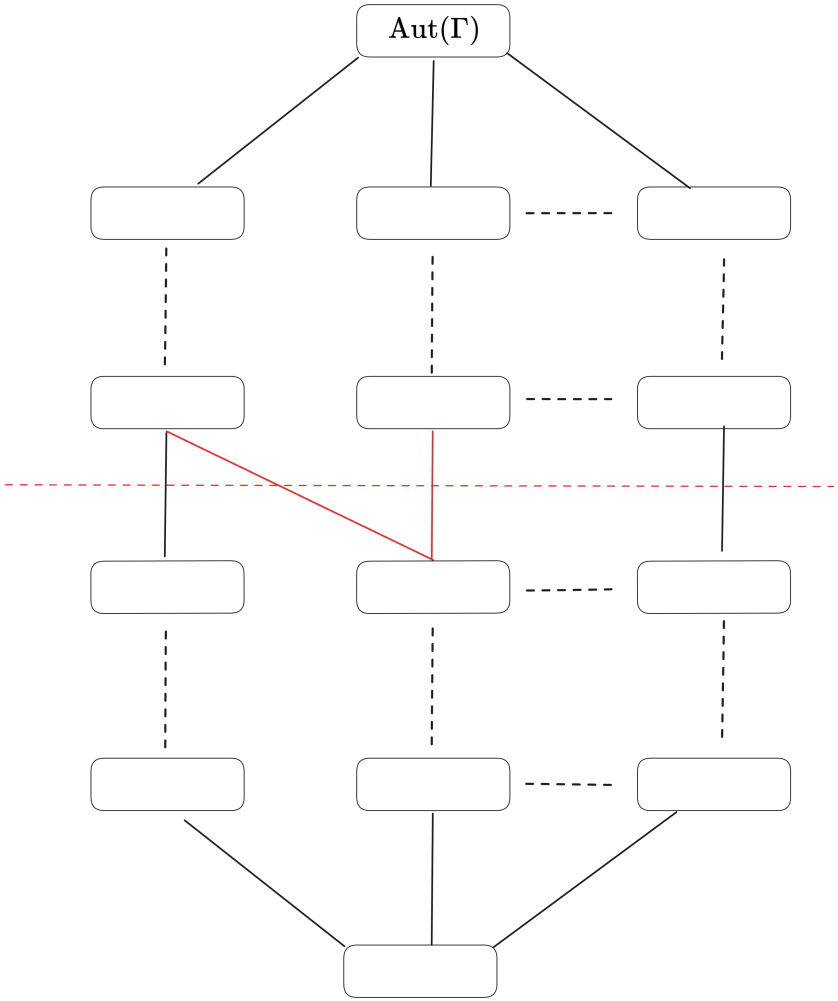
The **asymmetric depth** of a graph Γ of order n is $d(\Gamma) = n - k_{max}$

Definition

The **level of symmetry** of a graph Γ of order n is the ratio between the largest rank of a non-trivial partial automorphism of Γ and its order n.

- A graph Γ admitting a non-trivial automorphism has level of symmetry 1
- The level of symmetry of Γ is equal to the level of symmetry of its complement

Asymmetric depth



$\rightarrow$ rank n

$\vdots$

$\rightarrow$ rank $n - i$

$\vdots$

$\rightarrow$ rank k_{max}

Asymmetric depth & Symmetry Level of a Graph

Questions:

- Given $k \geq 1$, does there exist a graph Γ of order n with level of symmetry equal to $\frac{n-k}{n}$?
- What is the minimal level of symmetry of a graph Γ of order n as a function of n?
- ...

Asymmetric depth & Symmetry Level of a Graph

Questions:

- Given $k \geq 1$, does there exist a graph Γ of order n with level of symmetry equal to $\frac{n-k}{n}$?
- What is the minimal level of symmetry of a graph Γ of order n as a function of n?
- ...

(Gál, Mériová, Cingel, Pastorek, 2023+)
Searches, tests, computational results, bounds,...

Is there an upper bound for asymmetric depth $d = n - k_{max}$?

(with Jan Pastorek)

We proved that every n-vertex simple graph Γ admits a nontrivial partial automorphism on more than half of its vertices:

$$S(\Gamma) \geq 1 - \frac{n-1}{2n} > \frac{1}{2} \quad \text{equivalently} \quad d(\Gamma) \leq \frac{n-1}{2}.$$

Is there an upper bound for asymmetric depth $d = n - k_{max}$?

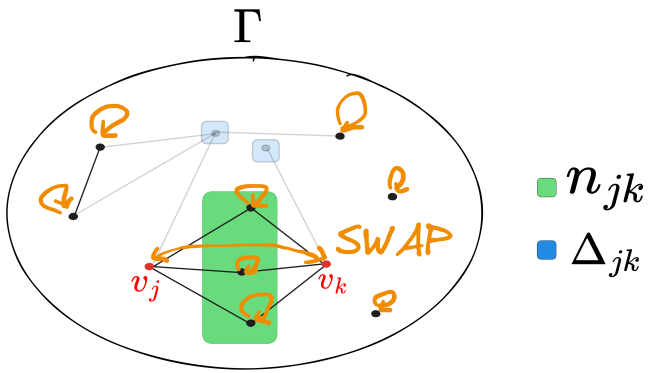
(with Jan Pastorek)

We proved that every n-vertex simple graph Γ admits a nontrivial partial automorphism on more than half of its vertices:

$$S(\Gamma) \geq 1 - \frac{n-1}{2n} > \frac{1}{2} \quad \text{equivalently} \quad d(\Gamma) \leq \frac{n-1}{2}.$$

Idea of the proof:

- Define a partial automorphism that will *swap just two vertices*
- Δ_{jk} - symmetric difference of the neighborhoods of v_j and v_k
- For any vertices $v_j \neq v_k$



This bound is sharp

Theorem

The graph Γ with 37 vertices with the following graph6 string representation meets the general bound

```
d~~vVB`pTErceQcmqK[YlafME|CLLZHTTWplTc[ie[
    ViMopRcqpQwNHhFyIqMXirMFJXKuBtFXipSmPdYozKkiVpbIxWcm`UclwXcmjLOjFRpIsl
```

Theorem

The graph Γ from the previous theorem with 37 vertices is of the smallest possible order that meets the bound.

This bound is sharp

- Computationally difficult - in both time and memory
- Suprisingly, the "most asymmetric" graph meets many regularitu conditions and it is a strongly regular graphs with parameters: $(n, \frac{n-1}{2}, \frac{n-5}{4}, \frac{n-1}{4}) \rightarrow$ conference graphs
- Spence's database of all conference graphs up to order 37 was used

This bound is sharp

- Computationally difficult - in both time and memory
- Suprisingly, the "most asymmetric" graph meets many regularitu conditions and it is a strongly regular graphs with parameters: $(n, \frac{n-1}{2}, \frac{n-5}{4}, \frac{n-1}{4}) \rightarrow$ conference graphs
- Spence's database of all conference graphs up to order 37 was used

- more in the next talk

Isomorphism Problem for Asymmetric Graphs

- In view of the prevalence of asymmetric graphs within the class of all graphs and the fact that graphs 'causing problems' to some of the most widely used algorithms are usually highly structured, it is interesting to restrict some of the isomorphism questions to asymmetric graphs, i.e., to consider the Asymmetric Graph Isomorphism

  Problem of deciding whether two given asymmetric graphs are isomorphic

Isomorphism Problem for Asymmetric Graphs

- In view of the prevalence of asymmetric graphs within the class of all graphs and the fact that graphs 'causing problems' to some of the most widely used algorithms are usually highly structured, it is interesting to restrict some of the isomorphism questions to asymmetric graphs, i.e., to consider the Asymmetric Graph Isomorphism

  Problem of deciding whether two given asymmetric graphs are isomorphic

- Specifically, one could ask whether deciding the Asymmetric Graph Isomorphism Problem might be easier than the general Graph Isomorphism Problem

What is easier?

(Schweitzer, Arvind, Das, Mukhopadhya, 2017)
For tournaments finding all symmetries and detecting asymmetry are polynomially equivalent.

Isomorphism Problem for Asymmetric graphs

- The open part of this question is whether the Graph Isomorphism Problem can be polynomially reduced to the Asymmetric Graph Isomorphism Problem

Isomorphism Problem for Asymmetric graphs

- The open part of this question is whether the Graph Isomorphism Problem can be polynomially reduced to the Asymmetric Graph Isomorphism Problem

- A possible way to answering this problem would lead through designing a polynomial complexity (in the order of the input graph) construction that would associate an asymmetric graph $\tilde{\Gamma}$ to each graph Γ while satisfying the property:

$$\tilde{\Gamma} \cong \tilde{\Gamma'} \text{ if and only if } \Gamma \cong \Gamma'$$

Isomorphism Problem for Asymmetric graphs

- Using the asymmetry level of a graph might allow for a further refinement of this question where one could ask whether the Asymmetric Graph Isomorphism Problem becomes easier with an increasing level of asymmetry of the considered graphs

Fixed Point Free Automorphism Problem

- Another complexity problem that appears to carry some relevance with regard to the above problems is the

  Fixed Point Free Automorphism Problem:
  Given a graph Γ, determine whether there exists a fixed-point-free automorphism of Γ
  (automorphism φ for which $\varphi(u) \neq u$, for all $u \in V(\Gamma)$)

Fixed Point Free Automorphism Problem

- Another complexity problem that appears to carry some relevance with regard to the above problems is the

  Fixed Point Free Automorphism Problem:
  Given a graph Γ, determine whether there exists a fixed-point-free automorphism of Γ
  (automorphism φ for which $\varphi(u) \neq u$, for all $u \in V(\Gamma)$)

- surprisingly, this problem has been shown to be NP-complete by Anna Lubiw in 1981

Fixed Point Free Automorphism Problem

- Another complexity problem that appears to carry some relevance with regard to the above problems is the

  Fixed Point Free Automorphism Problem:
  Given a graph Γ, determine whether there exists a fixed-point-free automorphism of Γ
  (automorphism φ for which $\varphi(u) \neq u$, for all $u \in V(\Gamma)$)

- surprisingly, this problem has been shown to be NP-complete by Anna Lubiw in 1981

- and remains NP-complete even when restricted to split, bipartite, k-subdivided and H-free graphs (provided H is not an induced subgraph of P_4) by the resent results of Aida Abiad et al. in 2026+

Advocacy for studying $PAut(\Gamma)$

- partial automorphisms capture local structure and local symmetries of a graph
- Babai (2018) argued that solving Graph Isomorphism Problem (GI) will involve looking at local structures of graphs (and so possibly partial automorphisms)
- the fastest procedure that helps to decide Graph Isomorphism problem (Weisfeiler-Leman algorithm) is based on the idea of local symmetries and aggregation of local information
- distinguishes structures inaccessible to group theory

Structure of $PAut(\Gamma)$ - inspiration from groups

Which groups are $Aut(\Gamma)$?

Theorem (Frucht 1939)

For any finite group G there exists a graph Γ such that $Aut(\Gamma) \cong G$.

When is an inverse monoid of partial permutations the partial automorphism monoid of a graph?

Situation in Inverse monoids is much more restrictive:

Theorem (R.Jajcay, T.J., N. Szakács, M. Szendrei 2021)

Given an inverse submonoid $S \leq \mathrm{PSym}(X)$, where X is a finite set, there exists a graph with vertex set X whose partial automorphism monoid is S if and only if the following conditions hold:

1. *S is a full inverse submonoid of $\mathrm{PSym}(X)$,*
2. *for any compatible subset $A \subseteq S$ of rank 1 partial permutations, if S contains the join of any two elements of A, then S contains the join of the set A,*
3. *the rank 2 elements of S form at most two $\mathcal{D}$-classes,*
4. *the $\mathcal{H}$-classes of rank 2 elements are nontrivial.*

When is an (abstract) inverse monoid *isomorphic* to the partial automorphism monoid of a graph?

Theorem (R.Jajcay, T.J., N. Szakács, M. Szendrei 2021)

Given a finite inverse monoid S, there exists a finite graph whose partial automorphism monoid is isomorphic to S if and only if the following conditions hold:

1. *S is Boolean,*
2. *S is fundamental,*
3. *for any subset $A \subseteq S$ of compatible 0-minimal elements, if all 2-element subsets of A have a join in S, then the set A has a join in S,*
4. *S has at most two $\mathcal{D}$-classes of height 2,*
5. *the $\mathcal{H}$-classes of the height 2 $\mathcal{D}$-classes of S are nontrivial.*

Algorithmic aspects of determining $PAut(\Gamma)$

- Computationally, one of the big problems in determining $Aut(\Gamma)$ is the part where one needs to prove that the list of automorphisms already found is complete.

Algorithmic aspects of determining $PAut(\Gamma)$

- Computationally, one of the big problems in determining $Aut(\Gamma)$ is the part where one needs to prove that the list of automorphisms already found is complete.
- Given an inverse monoid of partial automorphisms of Γ, if one wishes to determine whether it is the complete $PAut(\Gamma)$, one should ask whether it satisfies the "JJSS" characterization

Algorithmic aspects of determining $PAut(\Gamma)$

- Computationally, one of the big problems in determining $Aut(\Gamma)$ is the part where one needs to prove that the list of automorphisms already found is complete.
- Given an inverse monoid of partial automorphisms of Γ, if one wishes to determine whether it is the complete $PAut(\Gamma)$, one should ask whether it satisfies the "JJSS" characterization
- A partial automorphism φ of Γ is **complete** if there exists no partial automorphism ψ of Γ that is an extension of φ

Constructing PAut(Γ) recursively

Let $\Gamma = (V, E)$ be a graph,

- the results on symmetry levels of arbitrary graphs prove the existence of complete or extendable partial automorphisms of rank at least $\frac{n+1}{2n}$ for all graphs of order n

Constructing PAut(Γ) recursively

Let $\Gamma = (V, E)$ be a graph,

- the results on symmetry levels of arbitrary graphs prove the existence of complete or extendable partial automorphisms of rank at least $\frac{n+1}{2n}$ for all graphs of order n
- if $\varphi \in PSym(V)$ is a partial permutation of rank at least 2, then $\varphi \in PAut(\Gamma)$ if and only if $\varphi|_Y \in PAut(\Gamma)$ for any 2-element subset Y of dom φ, which means that we can iteratively build the whole inverse monoid of a graph from its partial automorphisms of rank 2

Constructing PAut(Γ) recursively

- we say that two partial automorphisms $\varphi_1, \varphi_2 \in PAut(\Gamma)$ are *compatible* if they are both restrictions of some partial automorphism $\psi \in PAut(\Gamma)$, i.e., if they agree on dom $\varphi_1 \cap$ dom φ_2 and have no shared image vertices on the rests of their domains

Constructing PAut(Γ) recursively

- we say that two partial automorphisms $\varphi_1, \varphi_2 \in PAut(\Gamma)$ are *compatible* if they are both restrictions of some partial automorphism $\psi \in PAut(\Gamma)$, i.e., if they agree on dom $\varphi_1 \cap$ dom φ_2 and have no shared image vertices on the rests of their domains
- thus, two partial automorphisms $\varphi_1, \varphi_2 \in PAut(\Gamma)$ of rank 2 are compatible if and only if both their domains and ranges are disjoint or their domains share a vertex on which they agree and the images of their non-shared domain vertices are different (we shall call the checking for this property a *compatibility comparison*)

Algorithm for constructing PAut(Γ) recursively

These observations yield the following algorithm:

Algorithm for constructing PAut(Γ) recursively

These observations yield the following algorithm:

1. initialize by constructing the empty automorphism ϵ and rank1 partial automorphisms $\varphi : u \mapsto v$ for all pairs of vertices u, v

Algorithm for constructing PAut(Γ) recursively

These observations yield the following algorithm:

1. initialize by constructing the empty automorphism ϵ and rank1 partial automorphisms $\varphi : u \mapsto v$ for all pairs of vertices u, v
2. for each pair of edges $\{u, v\}, \{u', v'\}$ construct rank 2 automorphisms mapping $u \mapsto u'$ and $v \mapsto v'$ and similarly for all pairs of pairs of non-adjacent vertices; these are *all* rank 2 partial automorphisms

Algorithm for constructing PAut(Γ) recursively

These observations yield the following algorithm:

1. initialize by constructing the empty automorphism ϵ and rank1 partial automorphisms $\varphi : u \mapsto v$ for all pairs of vertices u, v
2. for each pair of edges $\{u, v\}, \{u', v'\}$ construct rank 2 automorphisms mapping $u \mapsto u'$ and $v \mapsto v'$ and similarly for all pairs of pairs of non-adjacent vertices; these are *all* rank 2 partial automorphisms
3. having all rank i partial automorphisms of Γ, combine each of them with all compatible partial automorphisms of rank 2 to construct all partial automorphisms of rank $i + 1$

Algorithm for constructing PAut(Γ) recursively

These observations yield the following algorithm:

1. initialize by constructing the empty automorphism ϵ and rank1 partial automorphisms $\varphi : u \mapsto v$ for all pairs of vertices u, v
2. for each pair of edges $\{u, v\}, \{u', v'\}$ construct rank 2 automorphisms mapping $u \mapsto u'$ and $v \mapsto v'$ and similarly for all pairs of pairs of non-adjacent vertices; these are *all* rank 2 partial automorphisms
3. having all rank i partial automorphisms of Γ, combine each of them with all compatible partial automorphisms of rank 2 to construct all partial automorphisms of rank $i + 1$
4. stop when no partial automorphisms of rank i are compatible with any rank 2 partial automorphisms (i.e., when all partial automorphisms of rank i are complete)

Algorithm for constructing PAut(Γ) recursively

These observations yield the following algorithm:

1. initialize by constructing the empty automorphism ϵ and rank1 partial automorphisms $\varphi : u \mapsto v$ for all pairs of vertices u, v
2. for each pair of edges $\{u, v\}, \{u', v'\}$ construct rank 2 automorphisms mapping $u \mapsto u'$ and $v \mapsto v'$ and similarly for all pairs of pairs of non-adjacent vertices; these are *all* rank 2 partial automorphisms
3. having all rank i partial automorphisms of Γ, combine each of them with all compatible partial automorphisms of rank 2 to construct all partial automorphisms of rank $i + 1$
4. stop when no partial automorphisms of rank i are compatible with any rank 2 partial automorphisms (i.e., when all partial automorphisms of rank i are complete)
5. alternately, to obtain all partial automorphisms of rank $i = i' + i''$ from all rank i' and rank i'' partial automorphisms, combine each compatible pair into a rank $i = i' + i''$ partial automorphism

Computational complexity of the algorithm

- the algorithm constructs the entire $\mathrm{PAut}(\Gamma)$ in at most $n = |V(\Gamma)|$ steps, each of which consists of $|(\mathrm{PAut}(\Gamma))_i| \cdot |(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons (where $|(\mathrm{PAut}(\Gamma))_i|$ stands for the number of partial automorphisms of Γ of rank i)

Computational complexity of the algorithm

- the algorithm constructs the entire $\mathrm{PAut}(\Gamma)$ in at most $n = |V(\Gamma)|$ steps, each of which consists of $|(\mathrm{PAut}(\Gamma))_i| \cdot |(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons (where $|(\mathrm{PAut}(\Gamma))_i|$ stands for the number of partial automorphisms of Γ of rank i)
- using the alternate step, the algorithm constructs the automorphism group of Γ in $O(\log_2 n)$ steps each requiring $|(\mathrm{PAut}(\Gamma))_{i/2}| \cdot |(\mathrm{PAut}(\Gamma))_{i/2}|$ compatibility comparisons

Computational complexity of the algorithm

- the algorithm constructs the entire $\mathrm{PAut}(\Gamma)$ in at most $n = |V(\Gamma)|$ steps, each of which consists of $|(\mathrm{PAut}(\Gamma))_i| \cdot |(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons (where $|(\mathrm{PAut}(\Gamma))_i|$ stands for the number of partial automorphisms of Γ of rank i)
- using the alternate step, the algorithm constructs the automorphism group of Γ in $O(\log_2 n)$ steps each requiring $|(\mathrm{PAut}(\Gamma))_{i/2}| \cdot |(\mathrm{PAut}(\Gamma))_{i/2}|$ compatibility comparisons
- Determining whether a partial automorphism is complete and forming all its extensions of rank increased by 1 requires $|(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons

Computational complexity of the algorithm

- the algorithm constructs the entire $\mathrm{PAut}(\Gamma)$ in at most $n = |V(\Gamma)|$ steps, each of which consists of $|(\mathrm{PAut}(\Gamma))_i| \cdot |(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons (where $|(\mathrm{PAut}(\Gamma))_i|$ stands for the number of partial automorphisms of Γ of rank i)
- using the alternate step, the algorithm constructs the automorphism group of Γ in $O(\log_2 n)$ steps each requiring $|(\mathrm{PAut}(\Gamma))_{i/2}| \cdot |(\mathrm{PAut}(\Gamma))_{i/2}|$ compatibility comparisons
- Determining whether a partial automorphism is complete and forming all its extensions of rank increased by 1 requires $|(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons
- The problem of deciding whether a given partial automorphism can be extended into a (full) automorphism is polynomially equivalent to the Graph Isomorphism Problem

Computational complexity of the algorithm

- the algorithm constructs the entire $\mathrm{PAut}(\Gamma)$ in at most $n = |V(\Gamma)|$ steps, each of which consists of $|(\mathrm{PAut}(\Gamma))_i| \cdot |(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons (where $|(\mathrm{PAut}(\Gamma))_i|$ stands for the number of partial automorphisms of Γ of rank i)
- using the alternate step, the algorithm constructs the automorphism group of Γ in $O(\log_2 n)$ steps each requiring $|(\mathrm{PAut}(\Gamma))_{i/2}| \cdot |(\mathrm{PAut}(\Gamma))_{i/2}|$ compatibility comparisons
- Determining whether a partial automorphism is complete and forming all its extensions of rank increased by 1 requires $|(\mathrm{PAut}(\Gamma))_2|$ compatibility comparisons
- The problem of deciding whether a given partial automorphism can be extended into a (full) automorphism is polynomially equivalent to the Graph Isomorphism Problem
- Similarly, the problem of finding a full automorphism for a given restriction (if it exists) is also polynomially equivalent to the Graph Isomorphism Problem

Orbits of the automorphism group

The algorithm can also be used to compute the *pre-orbits* (unions of orbits) of the action of the automorphism group of Γ on $V(\Gamma)$:

- for each $(\mathrm{PAut}(\Gamma))_i$, define on $V(\Gamma)$ an equivalence relation $\sim_i$ by setting $u \sim_i v$ if and only if there exists a $\varphi \in (\mathrm{PAut}(\Gamma))_i$ mapping u to v

Orbits of the automorphism group

The algorithm can also be used to compute the *pre-orbits* (unions of orbits) of the action of the automorphism group of Γ on $V(\Gamma)$:

- for each $(\mathrm{PAut}(\Gamma))_i$, define on $V(\Gamma)$ an equivalence relation $\sim_i$ by setting $u \sim_i v$ if and only if there exists a $\varphi \in (\mathrm{PAut}(\Gamma))_i$ mapping u to v
- then the following are true:
  - if there exists an automorphism of Γ mapping u to v (i.e., u and v belong to the same orbit of the automorphism group of Γ on its vertices), then $u \sim_i v$ for all $1 \leq i \leq |V(\Gamma)|$

Orbits of the automorphism group

The algorithm can also be used to compute the *pre-orbits* (unions of orbits) of the action of the automorphism group of Γ on $V(\Gamma)$:

- for each $(\mathrm{PAut}(\Gamma))_i$, define on $V(\Gamma)$ an equivalence relation $\sim_i$ by setting $u \sim_i v$ if and only if there exists a $\varphi \in (\mathrm{PAut}(\Gamma))_i$ mapping u to v
- then the following are true:
  - if there exists an automorphism of Γ mapping u to v (i.e., u and v belong to the same orbit of the automorphism group of Γ on its vertices), then $u \sim_i v$ for all $1 \leq i \leq |V(\Gamma)|$
  - hence, if their exists an i, $2 \leq i \leq |V(\Gamma)|$, for which u and v are not $\sim_i$ equivalent, then u and v belong to different orbits of the automorphism group of Γ

Orbits of the automorphism group

The algorithm can also be used to compute the *pre-orbits* (unions of orbits) of the action of the automorphism group of Γ on $V(\Gamma)$:

- for each $(\mathrm{PAut}(\Gamma))_i$, define on $V(\Gamma)$ an equivalence relation $\sim_i$ by setting $u \sim_i v$ if and only if there exists a $\varphi \in (\mathrm{PAut}(\Gamma))_i$ mapping u to v
- then the following are true:
  - if there exists an automorphism of Γ mapping u to v (i.e., u and v belong to the same orbit of the automorphism group of Γ on its vertices), then $u \sim_i v$ for all $1 \leq i \leq |V(\Gamma)|$
  - hence, if their exists an i, $2 \leq i \leq |V(\Gamma)|$, for which u and v are not $\sim_i$ equivalent, then u and v belong to different orbits of the automorphism group of Γ
  - equivalently, the equivalence classes of each $\sim_i$ are unions of the orbits of the automorphism group of Γ on $V(\Gamma)$

Weisfeiler-Leman algorithm

Recall that determining the orbits of $Aut(\Gamma)$ is known to be polynomially equivalent to determining $Aut(\Gamma)$.

Weisfeiler-Leman algorithm

Recall that determining the orbits of $Aut(\Gamma)$ is known to be polynomially equivalent to determining $Aut(\Gamma)$.
One of the best known algorithms addressing the problem of determining the orbits of $Aut(\Gamma)$ is the *Weisfeiler-Leman algorithm*:

Weisfeiler-Leman algorithm

Recall that determining the orbits of $Aut(\Gamma)$ is known to be polynomially equivalent to determining $Aut(\Gamma)$.
One of the best known algorithms addressing the problem of determining the orbits of $Aut(\Gamma)$ is the *Weisfeiler-Leman algorithm*:

- The Weisfeiler-Leman algorithm is also used to determine the pre-orbits of $Aut(\Gamma)$
- If no changes occurred in the refinement step, the pre-orbits of $Aut(\Gamma)$ are determined by different colors

Weisfeiler-Leman algorithm

Recall that determining the orbits of $Aut(\Gamma)$ is known to be polynomially equivalent to determining $Aut(\Gamma)$.
One of the best known algorithms addressing the problem of determining the orbits of $Aut(\Gamma)$ is the *Weisfeiler-Leman algorithm*:

- The Weisfeiler-Leman algorithm is also used to determine the pre-orbits of $Aut(\Gamma)$
- If no changes occurred in the refinement step, the pre-orbits of $Aut(\Gamma)$ are determined by different colors
- (k-WL) can be computed for different values of k (with each higher value of k giving a refinement of the previous pre-orbits, until the pre-orbits become stable)
- Stable coloring by k-WL can be computed in time $O(n^{k+1} \log n)$

Weisfeiler-Leman algorithm

- k-WL **distinguishes** G and H if there is a colour c of the stable colouring such that G and H have different numbers of k-tuples of colour c.
- k-WL **identifies** G if it distinguishes G from all other graphs.

Remarks on k-WL

- The higher the k, the more graphs k-WL identifies

Remarks on k-WL

- The higher the k, the more graphs k-WL identifies
- For every k, there are graphs, which k-WL does not identify.

Remarks on k-WL

- The higher the k, the more graphs k-WL identifies
- For every k, there are graphs, which k-WL does not identify.
- Interesting (for theoretical and practical reasons) cases are for $k \leq 3$:
  - Used in practice
  - 1-WL identifies all forests (Immerman, Lander 1990).
  - 1-WL identifies almost all graphs (Babai, Erdös, Selkow 1980).
  - 1-WL and 2-WL do not distinguish any two regular graphs with the same number of vertices and the same number of edges.
  - 3-WL identifies almost all regular graphs (Bollobás 1982)
  - It is decidable in $O((n+m)logn)$ time whether 1-WL identifies a given graph (Kiefer et al., 2015)
  - Similar result for $k \geq 2$ is **unlikely** (Schweitzer, Kiefer, 2020)

Comparing the two algorithms

- both algorithms allow for computing pre-orbits of $Aut(\Gamma)$

Comparing the two algorithms

- both algorithms allow for computing pre-orbits of $Aut(\Gamma)$
- Calculations required for the i-WL and calculations required to compute $(\mathrm{PAut}(\Gamma))_i$ are somewhat comparable

Comparing the two algorithms

- both algorithms allow for computing pre-orbits of $Aut(\Gamma)$
- Calculations required for the i-WL and calculations required to compute $(\mathrm{PAut}(\Gamma))_i$ are somewhat comparable
- the complexity of i-WL increases with increasing i and then starts decreasing and so does the complexity of calculating $(\mathrm{PAut}(\Gamma))_i$

Comparing the two algorithms

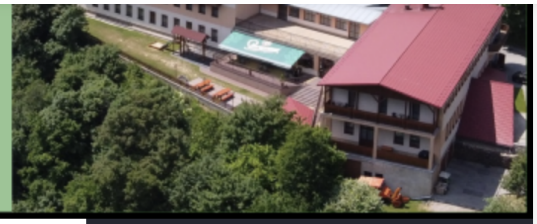
- both algorithms allow for computing pre-orbits of $Aut(\Gamma)$
- Calculations required for the i-WL and calculations required to compute $(\mathrm{PAut}(\Gamma))_i$ are somewhat comparable
- the complexity of i-WL increases with increasing i and then starts decreasing and so does the complexity of calculating $(\mathrm{PAut}(\Gamma))_i$
- both algorithms ultimately determine the orbits of $Aut(\Gamma)$; where stopping the algorithm using partial automorphisms before reaching the order of Γ not only determines that the automorphism group is trivial (and therefore each vertex is its own orbit), but also determines the level of symmetry of Γ

Thank you for listening!

Upcoming workshop in Slovakia

Upcoming workshop in Slovakia

ITAT Hotel Vršatec, Biele Karpaty



Invited speakers

Jorik Jooken (KU Leuven, Belgium): *How can computers help in obtaining new results in graph theory?*

This talk consists of two parts. In the first part, I will give an introduction to the domain of computer-assisted graph theory, which is concerned with developing algorithms in order to help researchers gain insights into various graph theoretical questions. In particular, I will discuss a broad range of popular techniques from this domain and briefly talk about their applications. In the second part, I will zoom in on an algorithm for exhaustively generating graphs and show how executing this algorithm leads to

Home page
Workshops proposals
Invited speakers
Workshops
Important dates
For authors
Paper submisison
Venue
Fees
Committees
Previous ITATs

https://itat.ics.upjs.sk/index.php?id=ws/CADM