

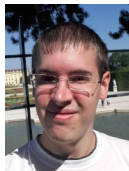


The Iteration Number of the Weisfeiler-Leman Algorithm

Martin Grohe (RWTH Aachen)

Moritz Lichter (TU Darmstadt)

Daniel Neuen (Bremen University)



Colour Refinement

The **colour refinement algorithm** iteratively computes a colouring of the vertices of a graph G . The colours represent local structural information about the vertices.

Colour Refinement

The **colour refinement algorithm** iteratively computes a colouring of the vertices of a graph G . The colours represent local structural information about the vertices.

Initialisation All vertices get the same colour.

Colour Refinement

The **colour refinement algorithm** iteratively computes a colouring of the vertices of a graph G . The colours represent local structural information about the vertices.

Initialisation All vertices get the same colour.

Refinement Step Vertices v, w get different colours if there is some colour c such that v and w have different numbers of neighbours of colour c .

Colour Refinement

The **colour refinement algorithm** iteratively computes a colouring of the vertices of a graph G . The colours represent local structural information about the vertices.

Initialisation All vertices get the same colour.

Refinement Step Vertices v, w get different colours if there is some colour c such that v and w have different numbers of neighbours of colour c .

Refinement is repeated until colouring stays **stable**.

Colour Refinement

The **colour refinement algorithm** iteratively computes a colouring of the vertices of a graph G . The colours represent local structural information about the vertices.

Initialisation All vertices get the same colour.

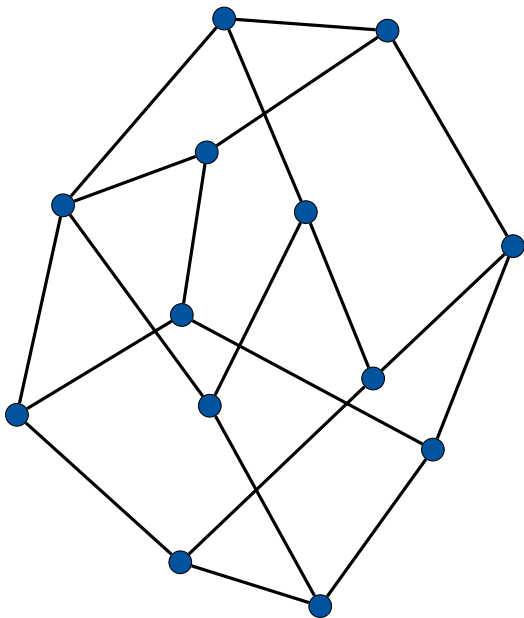
Refinement Step Vertices v, w get different colours if there is some colour c such that v and w have different numbers of neighbours of colour c .

Refinement is repeated until colouring stays **stable**.

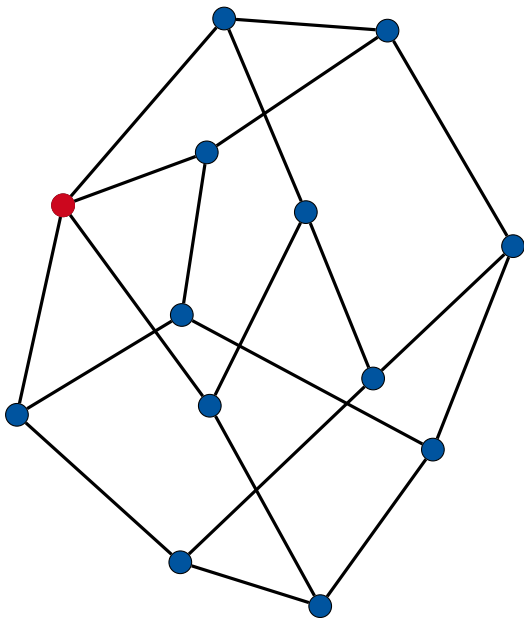
Remark

Colour refinement is essentially the same as the **1-dimensional Weisfeiler-Leman** algorithm.

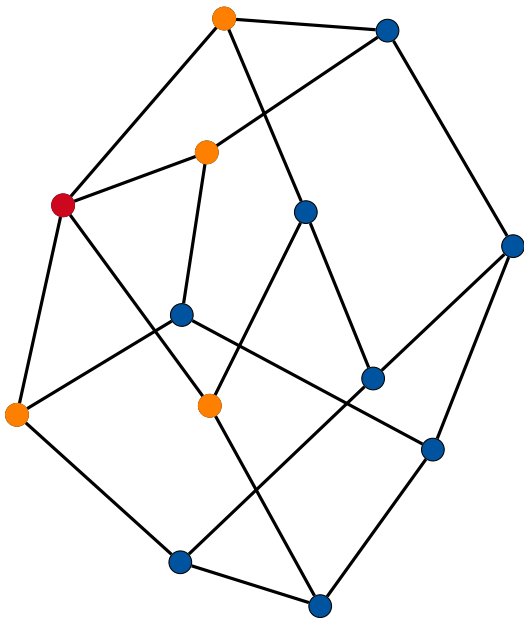
Example



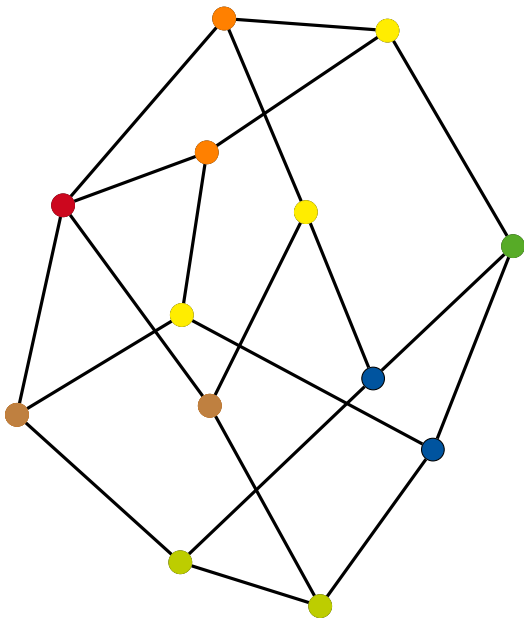
Example



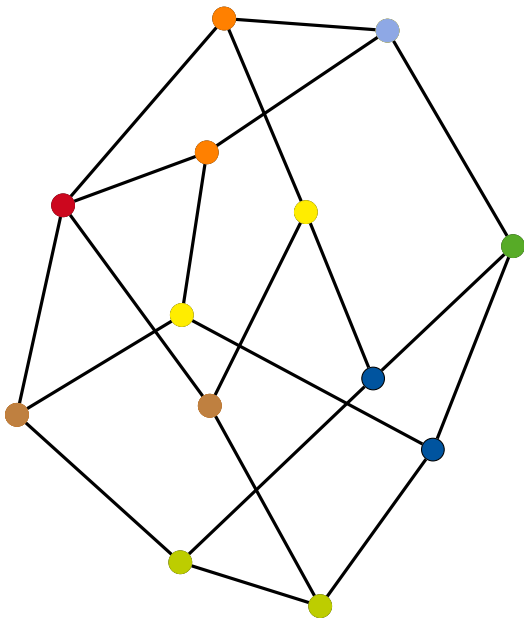
Example



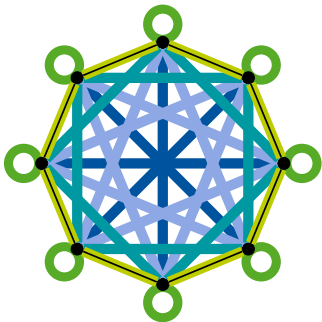
Example



Example

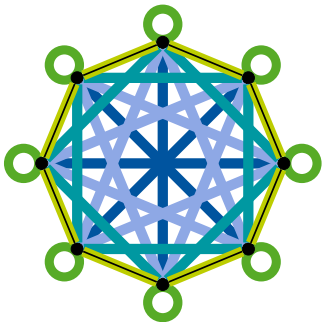


Higher-Dimensional Weisfeiler-Leman



The **Weisfeiler-Leman algorithm** generalises the idea of aggregating local structural information in a colouring from vertices to tuples of vertices.

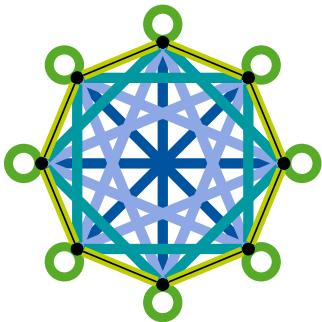
Higher-Dimensional Weisfeiler-Leman



The **Weisfeiler-Leman algorithm** generalises the idea of aggregating local structural information in a colouring from vertices to tuples of vertices.

k -WL iteratively colours k -tuples of vertices.

Higher-Dimensional Weisfeiler-Leman



The **Weisfeiler-Leman algorithm** generalises the idea of aggregating local structural information in a colouring from vertices to tuples of vertices.

k -WL iteratively colours k -tuples of vertices.

(Weisfeiler and Leman 1968,
Babai \sim 1980)

Atomic Types and Partial Isomorphisms

Let G, H be graphs and $\mathbf{v} = (v_1, \dots, v_k) \in V(G)^k$,
 $\mathbf{w} = (w_1, \dots, w_k) \in V(H)^k$.

Atomic Types and Partial Isomorphisms

Let G, H be graphs and $\mathbf{v} = (v_1, \dots, v_k) \in V(G)^k$,
 $\mathbf{w} = (w_1, \dots, w_k) \in V(H)^k$.

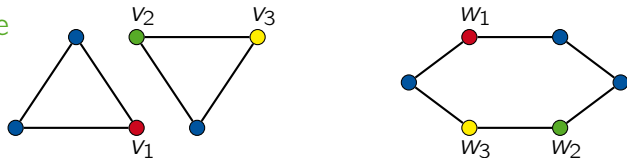
The **atomic type** $\text{atp}(\mathbf{v})$ of \mathbf{v} in G is the isomorphism type of the labelled induced subgraph $G[\{v_1, \dots, v_k\}]$.

Atomic Types and Partial Isomorphisms

Let G, H be graphs and $\mathbf{v} = (v_1, \dots, v_k) \in V(G)^k$,
 $\mathbf{w} = (w_1, \dots, w_k) \in V(H)^k$.

The **atomic type** $\text{atp}(\mathbf{v})$ of \mathbf{v} in G is the isomorphism type of the labelled induced subgraph $G[\{v_1, \dots, v_k\}]$.

Example



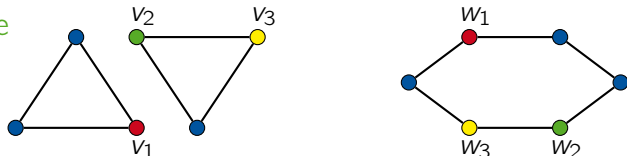
(v_1, v_2, v_3) and (w_1, w_2, w_3) have the same atomic type.

Atomic Types and Partial Isomorphisms

Let G, H be graphs and $\mathbf{v} = (v_1, \dots, v_k) \in V(G)^k$,
 $\mathbf{w} = (w_1, \dots, w_k) \in V(H)^k$.

The **atomic type** $\text{atp}(\mathbf{v})$ of \mathbf{v} in G is the isomorphism type of the labelled induced subgraph $G[\{v_1, \dots, v_k\}]$.

Example



(v_1, v_2, v_3) and (w_1, w_2, w_3) have the same atomic type.
So do (v_1, v_2, v_3, v_1) and (w_1, w_3, w_2, w_1) .

The Weisfeiler Leman Algorithm

Initial Colouring wl_0

$wl_0(\mathbf{v}) = wl_0(\mathbf{w})$ iff $atp(\mathbf{v}) = atp(\mathbf{w})$.

The Weisfeiler Leman Algorithm

Initial Colouring wl_0

$wl_0(\mathbf{v}) = wl_0(\mathbf{w})$ iff $atp(\mathbf{v}) = atp(\mathbf{w})$.

Refinement Step $wl_i \rightarrow wl_{i+1}$

$wl_{i+1}(\mathbf{v}) = wl_{i+1}(\mathbf{w})$ iff for all atomic types a and all colours c_1, \dots, c_k in the range of wl_i ,

$$\begin{array}{l} \#v \in V(G) \text{ such that} \\ atp(v_1, \dots, v_k, v) = a \\ wl_i(v, v_2, v_3 \dots, v_k) = c_1 \\ wl_i(v_1, v, v_3 \dots, v_k) = c_2 \\ \vdots \\ wl_i(v_1, \dots, v_{k-1}, v) = c_k \end{array} = \begin{array}{l} \#w \in V(G) \text{ such that} \\ atp(w_1, \dots, w_k, w) = a \\ wl_i(w, w_2, w_3 \dots, w_k) = c_1 \\ wl_i(w_1, w, w_3, \dots, w_k) = c_2 \\ \vdots \\ wl_i(w_1, \dots, w_{k-1}, w) = c_k \end{array}$$

The Weisfeiler Leman Algorithm

Initial Colouring wl_0

$wl_0(\mathbf{v}) = wl_0(\mathbf{w})$ iff $atp(\mathbf{v}) = atp(\mathbf{w})$.

Refinement Step $wl_i \rightarrow wl_{i+1}$

$wl_{i+1}(\mathbf{v}) = wl_{i+1}(\mathbf{w})$ iff for all atomic types a and all colours c_1, \dots, c_k in the range of wl_i ,

$$\begin{array}{l} \#v \in V(G) \text{ such that} \\ atp(v_1, \dots, v_k, v) = a \\ wl_i(v, v_2, v_3, \dots, v_k) = c_1 \\ wl_i(v_1, v, v_3, \dots, v_k) = c_2 \\ \vdots \\ wl_i(v_1, \dots, v_{k-1}, v) = c_k \end{array} = \begin{array}{l} \#w \in V(G) \text{ such that} \\ atp(w_1, \dots, w_k, w) = a \\ wl_i(w, w_2, w_3, \dots, w_k) = c_1 \\ wl_i(w_1, w, w_3, \dots, w_k) = c_2 \\ \vdots \\ wl_i(w_1, \dots, w_{k-1}, w) = c_k \end{array}$$

Refinement is repeated until colouring stays **stable**.

- ▶ WL was originally designed as a graph isomorphism heuristics

- ▶ WL was originally designed as a graph isomorphism heuristics
- ▶ It has characterisations in terms of logic, algebra, and graph neural networks.

- ▶ WL was originally designed as a graph isomorphism heuristics
- ▶ It has characterisations in terms of logic, algebra, and graph neural networks.
- ▶ Besides isomorphism testing, it has applications in combinatorial optimisation and machine learning.

What is the **iteration number** of k -WL?

That is, how many refinement rounds does k -WL need until it reaches a stable colouring?

n = number of vertices, k = dimension

Upper Bounds

- ▶ Trivial: $n^k - 1$
- ▶ Kiefer, Schweitzer 2017: $O\left(\frac{n^2}{\log n}\right)$ for $k = 2$
- ▶ Lichter, Ponomarenko, Schweitzer 2019: $O(n \log n)$ for $k = 2$
- ▶ G., Verbitsky 2006; G., Kiefer 2021, van Bergerem, G., Kiefer, Oeljeklaus 2023 (next talk): $O(\log n)$ iterations on bounded tree width, planar, interval graphs

n = number of vertices, k = dimension

Upper Bounds

- ▶ Trivial: $n^k - 1$
- ▶ Kiefer, Schweitzer 2017: $O\left(\frac{n^2}{\log n}\right)$ for $k = 2$
- ▶ Lichter, Ponomarenko, Schweitzer 2019: $O(n \log n)$ for $k = 2$
- ▶ G., Verbitsky 2006; G., Kiefer 2021, van Bergerem, G., Kiefer, Oeljeklaus 2023 (next talk): $O(\log n)$ iterations on bounded tree width, planar, interval graphs

Lower Bounds

- ▶ Fürer 2001: $\Omega(n)$
- ▶ Kiefer, McKay 2020: $n - 1$ for $k = 1$
- ▶ Berkholz, Nordström 2016: $n^{\Omega\left(\frac{k}{\log k}\right)}$ on k -ary relational structures

Theorem

The iteration number of k -WL is $O(kn^{k-1} \log n)$ on all relational structures.

Theorem

The iteration number of k -WL is $O(kn^{k-1} \log n)$ on all relational structures.

Proof idea.

Algebraic:

- ▶ Translate sequence of colourings to increasing sequence of finite-dimensional semi-simple algebras of $n^{k-1} \times n^{k-1}$ -matrices.
- ▶ Use known fact from representation theory that such a sequence cannot be much longer than the dimension of the matrices.

Based on (Lichter, Ponomarenko, Schweitzer 2019).



Theorem

Let $k' \geq k \geq 1$. For all sufficiently large n there are k -ary relational structures A, B of size n such that

- ▶ k -WL distinguishes A and B ;
- ▶ even k' -WL needs $n^{\Omega(k)}$ iterations to distinguish A, B .

Theorem

Let $k' \geq k \geq 1$. For all sufficiently large n there are k -ary relational structures A, B of size n such that

- ▶ k -WL distinguishes A and B ;
- ▶ even k' -WL needs $n^{\Omega(k)}$ iterations to distinguish A, B .

Proof idea.

Based on (Berkholz, Nordström 2016):

CFI-type construction combined with a compression technique from proof complexity due to (Razborov 2016). □

Theorem

Let $k' \geq k \geq 1$. For all sufficiently large n there are k -ary relational structures A, B of size n such that

- ▶ k -WL distinguishes A and B ;
- ▶ even k' -WL needs $n^{\Omega(k)}$ iterations to distinguish A, B .

Proof idea.

Based on (Berkholz, Nordström 2016):

CFI-type construction combined with a compression technique from proof complexity due to (Razborov 2016). □

Corollary

The iteration number of k -WL is $n^{\Omega(k)}$ on k -ary relational structures.

Recent Improvement

Theorem (G., Lichter, Neuen, Schweitzer 2023+)

The iteration number of k -WL is $\Omega(n^{k/2})$ on graphs.

Conjecture

The iteration number of k -WL is $\Omega(n^{k-1-o(1)})$ on graphs.