Hot Topics in Graph Neural Networks

Graphs in Artificial Intelligence and Neural Networks

Josephine Thomas, Silvia Beddar-Wiesing, Clara Holzhüter, Alice Moallemi-Oureh

25th of October 2022
The Research Center for Information System Design (ITeG) at the University of Kassel focuses on socially responsible IT design.

We promote responsible, socially sustainable digitisation through interdisciplinary research.

12 research groups from different disciplines (computer science, IT and privacy law, information systems, psychology, sociology, human-machine systems engineering).

Main areas of mutual research

- **Methods** of socio-technical IT Design to increase digital self-determination and sovereignty
- **Privacy** and the dynamics of the information society
- **AI and Hybrid Intelligence** and their embedding in the social system
The Team

Josephine Thomas  Silvia Beddar-Wiesing  Alice Moallemy-Oureh

Clara Holzhüter  Bernhard Sick  Christoph Scholz

Eric Alsmann
Rüdiger Nather
Till-Mattis Nebel
Björn-André Schröder
# Workshop Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00 - 11:15</td>
<td>GAIN</td>
<td>Expressivity and Dynamic of GNNs in theory and applications to the power grid</td>
</tr>
<tr>
<td>11:20 - 12:05</td>
<td>Petar Veličković</td>
<td>Algorithmically-aligned GNNs</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Lunch Break</strong></td>
</tr>
<tr>
<td>13:15 - 13:40</td>
<td>Fabian Jogl</td>
<td>Do we need to Improve Message Passing?</td>
</tr>
<tr>
<td>13:45 - 14:10</td>
<td>Maximilian Thiessen</td>
<td>Expectation Complete Graph Representations using Graph Homomorphisms</td>
</tr>
<tr>
<td>14:15 - 15:00</td>
<td>Massimo Perini</td>
<td>Graph Streams</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Coffee Break</strong></td>
</tr>
<tr>
<td>16:00 - 16:45</td>
<td>Antonio Longa</td>
<td>Explaining the explainers in GNNs: a comparative study</td>
</tr>
<tr>
<td>16:50 - 17:35</td>
<td>Hannes Stärk</td>
<td>Geometric ML for Molecules</td>
</tr>
</tbody>
</table>
1. Graph Neural Networks for different Graph Types: A Survey

2. The Modeling Power of different Graph Types

3. WL goes Dynamic: Expressivity of GNNs for Attributed and Dynamic Graphs

4. Extension of the WL Hierarchy by WL Tests for Arbitrary Graphs

5. Graph Neural Networks for Power Grids

6. Ongoing Research
Graph Neural Networks for different Graph Types: A Survey

What can GNNs achieve nowadays and where is work to be done?
What can GNNs achieve nowadays and where is work to be done?

- GNNs extend Neural Networks to work on graphs
What can GNNs achieve nowadays and where is work to be done?

- GNNs extend Neural Networks to work on graphs
- The architecture of GNNs can be different depending on the properties of a graph
What can GNNs achieve nowadays and where is work to be done?

- GNNs extend Neural Networks to work on graphs
- The architecture of GNNs can be different depending on the properties of a graph
- Graph properties yield graph types → Which graph types are there?
What can GNNs achieve nowadays and where is work to be done?

- GNNs extend Neural Networks to work on graphs
- The architecture of GNNs can be different depending on the properties of a graph
- Graph properties yield graph types → Which graph types are there?
- Which graph types can be handled by GNN models?
GNNs for different Graph Types: The graph types

- static undirected graph
- static structural properties
- semantic graph properties
- dynamic structural properties
GNNs for different Graph Types: The graph types

- static undirected graph
- static structural properties
- semantic graph properties
- dynamic structural properties

- directed hypergraph
GNNs for different Graph Types: The graph types

- static undirected graph
- static structural properties
- semantic graph properties
- dynamic structural properties

complete graph
GNNs for different Graph Types: The graph types

- static undirected graph
- static structural properties
- semantic graph properties
- dynamic structural properties

strictly shrinking graph
GNNs for different Graph Types: The representation of dynamic graphs
GNNs for different Graph Types: The representation of dynamic graphs

- discrete-time
- dynamic
GNNs for different Graph Types: The representation of dynamic graphs

- **discrete-time dynamic**

- **continuous-time dynamic**
GNNs for different Graph Types: Where are the gaps?
Much work has been done on GNN models for **static graphs**
GNNs for different Graph Types: Where are the gaps?

- Much work has been done on GNN models for **static graphs**
  - Exception: Multigraphs
GNNs for different Graph Types: Where are the gaps?

- Much work has been done on GNN models for **static graphs**
  - Exception: Multigraphs
- For **hypergraphs few models exist** for each graph type
Much work has been done on GNN models for static graphs
- Exception: Multigraphs

For hypergraphs few models exist for each graph type

For graphs and hypergraphs in discrete-time models exist similar to the static case
Much work has been done on GNN models for **static graphs**
- Exception: Multigraphs

For **hypergraphs few models exist** for each graph type

For **graphs and hypergraphs in discrete-time** models exist similar to the static case

Many gaps in models for **graphs in continuous-time**
Much work has been done on GNN models for **static graphs**
- Exception: Multigraphs
- For **hypergraphs few models exist** for each graph type
- For **graphs and hypergraphs in discrete-time** models exist similar to the static case
- Many gaps in models for **graphs in continuous-time**
  - Deletion of nodes/edges
Much work has been done on GNN models for static graphs
  - Exception: Multigraphs
For hypergraphs few models exist for each graph type
For graphs and hypergraphs in discrete-time models exist similar to the static case
Many gaps in models for graphs in continuous-time
  - Deletion of nodes/edges
  - Dynamic attributes, especially if data-type is complex
Models for **combined graph types** and **semantic graph properties** are developed application specific.
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific
- Some **semantic graph properties** are not explicitly handled
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific.
- Some **semantic graph properties** are not explicitly handled.
  - **unconnected graphs**
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific
- Some **semantic graph properties** are not explicitly handled
  - unconnected graphs
  - acyclic graphs
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific.

- Some **semantic graph properties** are not explicitly handled:
  - unconnected graphs
  - acyclic graphs
  - $r$-regular graphs
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific.
- Some **semantic graph properties** are not explicitly handled:
  - unconnected graphs
  - acyclic graphs
  - r-regular graphs
- Many gaps in models for **hypergraphs in continuous-time**
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific.
- Some **semantic graph properties** are not explicitly handled:
  - unconnected graphs
  - acyclic graphs
  - r-regular graphs
- Many gaps in models for **hypergraphs in continuous-time**
  - Node/edge-attributes
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific
- Some **semantic graph properties** are not explicitly handled
  - unconnected graphs
  - acyclic graphs
  - r-regular graphs
- Many gaps in models for **hypergraphs in continuous-time**
  - Node/edge-attributes
  - Node/edge-heterogenity
GNNs for different Graph Types: Where are the gaps?

- Models for **combined graph types** and **semantic graph properties** are developed application specific.
- Some **semantic graph properties** are not explicitly handled:
  - unconnected graphs
  - acyclic graphs
  - r-regular graphs
- Many gaps in models for **hypergraphs in continuous-time**:
  - Node/edge-attributes
  - Node/edge-heterogenity
  - Multiple nodes/edges
The Modeling Power of different Graph Types

How do we assess the ability of different graph types to represent information?
How do we assess the ability of different graph types to represent information?

- Expressivity relation
How do we assess the ability of different graph types to represent information?

- Expressivity relation
- Examples of expressivity relations
How do we assess the ability of different graph types to represent information?

- Expressivity relation
- Examples of expressivity relations
- All attributed graph types can be transformed into a SAUHG
The Modeling Power of different Graph Types

How do we assess the ability of different graph types to represent information?

- Expressivity relation
- Examples of expressivity relations
- All attributed graph types can be transformed into a SAUHG
- All attributed graph types are equally expressive
How do we assess the ability of different graph types to represent information?

A graph type $G_2$ is **at least as expressive** as a graph type $G_1$, if and only if $G_2$ encodes at least as many graph properties as $G_1$ denoted as $G_1 \preceq G_2$. In case both types encode the same graph properties it is denoted as $G_1 \approx G_2$. 
The Modeling Power of different Graph Types: Express. relation examples

Transformation directed to undirected graph:
Storing the directions and multiple attributes in the new attributes.
The Modeling Power of different Graph Types: Expres. relation examples

\[ \mathcal{G}_{\text{dynamic}} \preceq \mathcal{G}_{\text{static}} \]

**Transformation dynamic to static graph:**
Cumulating the structural information in one entire graph and storing the corresponding attribute time series as the new attributes.
The Modeling Power of different Graph Types: Express. relation examples

Transformation dynamic to static graph:
Cummulating the structural information in one entire graph and storing the corresponding attribute time series as the new attributes.
The Modeling Power of different Graph Types: Results
All attributed graph types can be transformed into a **static attributed undirected homogeneous graph (SAUHG)**.
All attributed graph types can be transformed into a static attributed undirected homogeneous graph (SAUHG).

All attributed graph types are equally expressive.
All attributed graph types can be transformed into a static attributed undirected homogeneous graph (SAUHG).

All attributed graph types are equally expressive.

→ We can transform graph data to be able to use an arbitrary GNN.
All attributed graph types can be transformed into a static attributed undirected homogeneous graph (SAUHG).

All attributed graph types are equally expressive.

→ We can transform graph data to be able to use an arbitrary GNN.
→ We are free to choose a graph type that models our problem best.
Weisfeiler–Lehmann goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs
Weisfeiler-Lehman goes dynamic
Motivation: Expressivity of GNNs
Weisfeiler-Lehman goes dynamic
Motivation: Expressivity of GNNs

Which graphs/nodes can a GNN distinguish?

Scarselli et. al (2009)
GNNs cannot distinguish nodes having the same unfolding trees.

Xu et. al (2018)
GNNs are as powerful as the Weisfeiler-Lehman graph isomorphism test (1-WL, 1968).

D’Inverno et. al (2021)
The WL-test and the unfolding trees induce the same equivalence relationship on graphs.
Weisfeiler-Lehman goes dynamic
Motivation: Expressivity of GNNs

Which **graphs/nodes** can a GNN **distinguish**?

**Scarselli et. al (2009)**
GNNs cannot distinguish nodes having the **same unfolding trees**.

**Xu et. al (2018)**
GNNs are **as powerful as** the Weisfeiler-Lehman graph isomorphism test (1-WL, 1968).

**D’Inverno et. al (2021)**
The WL-test and the unfolding trees induce the **same equivalence** relationship on graphs.

→ **static node-attributed graphs only!**
Weisfeiler-Lehman goes dynamic
Motivation: Expressivity of GNNs
Weisfeiler-Lehman goes dynamic
Motivation: Expressivity of GNNs

Which **functions** can a GNN **approximate**?

---

**D’Inverno et. al (2021)**
Message Passing GNNs can approximate in probability **any measurable function** that respects the unfolding equivalence.

**Azizian et. al (2020)**
Message Passing GNNs are **dense in continuous functions** on graphs modulo 1-WL.
Weisfeiler-Lehman goes dynamic
Motivation: Expressivity of GNNs

Which **functions** can a GNN **approximate**?

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Passing GNNs can <strong>approximate</strong> in probability any <strong>measurable function</strong> that respects the unfolding equivalence.</td>
<td>Message Passing GNNs are <strong>dense in continuous functions</strong> on graphs modulo 1-WL.</td>
</tr>
</tbody>
</table>

→ static node-attributed graphs only!
Weisfeiler-Lehman goes dynamic\textsuperscript{1}
Motivation: Expressivity of GNNs

Contributions

\textsuperscript{1}Beddar-Wiesing, D'Inverno, Graziani, Lachi, Moallemi-Oureh, Scarselli, Thomas: \textit{Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs}, arxiv preprint
Weisfeiler-Lehman goes dynamic\textsuperscript{1}
Motivation: Expressivity of GNNs

Contributions

- **Extension** of WL-Tests and unfolding trees to (edge-)attributes and dynamics
- **Proof of Extended Approximation Theorems**: GNNs can approximate to any precision and probability any measurable function on attributed and dynamic graphs

\textsuperscript{1}Beddar-Wiesing, D’Inverno, Graziani, Lachi, Moallemiy-Oureh, Scarselli, Thomas: *Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs*, arxiv preprint
Weisfeiler-Lehman goes dynamic
Recap: WL-Test and Unfolding Trees
Weisfeiler-Lehman goes dynamic
Recap: WL-Test and Unfolding Trees
Weisfeiler-Lehman goes dynamic
Recap: WL-Test and Unfolding Trees

![Graph and tree diagram showing the process of WL-test for different k values.](image)
Weisfeiler-Lehman goes dynamic
Recap: WL-Test and Unfolding Trees

GAIN
Weisfeiler-Lehman goes dynamic
Recap: WL-Test and Unfolding Trees

(Thank you Nils Kriege for the wonderful illustration!)
Recap: WL-Test and Unfolding Trees

Unfolding Trees

Unfolding Trees of both blue nodes
Weisfeiler-Lehman goes dynamic
Extension of WL-Test and Unfolding Trees

WL Coloring for Attributed Graphs

hashed node/edge attribute
attribute aggregation
Weisfeiler-Lehman goes dynamic
Extension of WL-Test and Unfolding Trees

Unfolding Trees for Attributed Graphs

Unfolding Trees of both blue nodes
Weisfeiler-Lehman goes dynamic
Extension of WL-Test and Unfolding Trees

WL Coloring for Dynamic Graphs
Weisfeiler-Lehman goes dynamic
Extension of WL-Test and Unfolding Trees

Unfolding Trees for Dynamic Graphs

<table>
<thead>
<tr>
<th>Dynamic Graph G</th>
<th>Statified Graph G'</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Dynamic Graph G" /></td>
<td><img src="image2" alt="Statified Graph G'" /></td>
</tr>
</tbody>
</table>

- Dynamic Graph G
- Statified Graph G'

\[
\begin{align*}
\text{Dynamic Graph G} & \quad \text{Statified Graph G'} \\
\text{Time} & \quad \text{Time} \\
\text{Dynamic unfolding trees of } a, c & \quad \text{Statified unfolding trees of } a, c
\end{align*}
\]
Weisfeiler-Lehman goes dynamic
Equivalence of WL and UT

Proposition
For all nodes \( u, v \) holds:

1. in the attributed case:
   \[ u \sim_{\text{AWL}} v \iff u \sim_{\text{AUT}} v. \]

2. in the dynamic case:
   \[ u \sim_{\text{DWL}} v \iff u \sim_{\text{DUT}} v. \]
Weisfeiler-Lehman goes dynamic
Generic GNNs: GNN for SAUHG (SGNN) and dynamic graphs (MP-DGNN)
Weisfeiler-Lehman goes dynamic

Generic GNNs: GNN for SAUHGs (SGNN) and dynamic graphs (MP-DGNN)

For a SAUHG $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$, let $v \in \mathcal{V}$. The propagation scheme of the SGNN for one iteration $k \in [K]$ is defined as

$$h_v^k = \text{COMBINE} \left( h_v^{k-1}, \text{AGGREGATE} \left( \{ h_u^{k-1} \}_{u \in \mathcal{N}(v)}, \{ \omega(\{u, v\}) \}_{u \in \mathcal{N}(v)} \right) \right).$$
Weisfeiler-Lehman goes dynamic

Generic GNNs: GNN for SAUHGs (SGNN) and dynamic graphs (MP-DGNN)

For a SAUHG $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$, let $v \in \mathcal{V}$. The propagation scheme of the SGNN for one iteration $k \in [K]$ is defined as

$$h^k_v = \text{COMBINE} \left( h^{k-1}_v, \text{AGGREGATE} \left( \{h^{k-1}_u\}_{u \in \mathcal{N}(v)}, \{\omega(u, v)\}_{u \in \mathcal{N}(v)} \right) \right).$$

For a discrete dynamic graph $G' = (G_t)_{t \in I}$, let $v \in \mathcal{V}_t$. The propagation scheme of the MP-DGNN for one iteration $k \in [K]$ at timestamp $t \in [T]$ is defined as

$$h^k_v(t) = \text{COMBINE}^{(k)}_t \left( h^{k-1}_v(t), \text{AGGREGATE}^{k}_t \left( \{h^{k-1}_u(t)\}_{u \in \mathcal{N}_t(v)}, \{\omega(u, v)(t)\}_{u \in \mathcal{N}_t(v)} \right) \right).$$
Weisfeiler-Lehman goes dynamic

Universal Approximation of SGNN and MP-DGNN
Weisfeiler-Lehman goes dynamic
Universal Approximation of SGNN and MP-DGNN

For

- Domain of SAUHGs $\mathcal{G}$ and
  $r = \max_{g \in \mathcal{G}} \text{diam}(G);$  
- any measurable function $f$ preserving
  $\sim_{\text{AUT}};$  
- any norm $\| \cdot \|$ on $\mathbb{R}$ and probability
  measure $P$ on $\mathcal{G};$  
- $\epsilon, \lambda \in \mathbb{R}$, precision $\epsilon > 0$, probability
  $\lambda \in (0, 1).$
Weisfeiler-Lehman goes dynamic
Universal Approximation of SGNN and MP-DGNN

For

- Domain of SAUHGs $\mathcal{G}$ and $r = \max_{g \in \mathcal{G}} \text{diam}(G)$;
- any measurable function $f$ preserving $\sim_{\text{AUT}}$;
- any norm $\| \cdot \|$ on $\mathbb{R}$ and probability measure $P$ on $\mathcal{G}$;
- $\epsilon, \lambda \in \mathbb{R}$, precision $\epsilon > 0$, probability $\lambda \in (0, 1)$.

There exists an SGNN s.t. the function $\varphi$ realized by the SGNN, computed after $r + 1$ steps for all $G \in \mathcal{G}$ and $v \in G$, satisfies:

$$P \left( \| f(G, v) - \varphi(G, v) \| \leq \epsilon \right) \geq 1 - \lambda.$$
Weisfeiler-Lehman goes dynamic
Universal Approximation of SGNN and MP-DGNN

For
- Domain of SAUHGs $\mathcal{G}$ and $r = \max_{G \in \mathcal{G}} \text{diam}(G)$;
- any measurable function $f$ preserving $\sim_{\text{AUT}}$;
- any norm $\| \cdot \|$ on $\mathbb{R}$ and probability measure $P$ on $\mathcal{G}$;
- $\epsilon, \lambda \in \mathbb{R}$, precision $\epsilon > 0$, probability $\lambda \in (0, 1)$.

There exists an SGNN s.t. the function $\varphi$ realized by the SGNN, computed after $r + 1$ steps for all $G \in \mathcal{G}$ and $v \in G$, satisfies:

$$P \left( \| f(G, v) - \varphi(G, v) \| \leq \epsilon \right) \geq 1 - \lambda.$$

For
- Domain of discrete dyn. graphs $G' = (G_t)_{t \in I} \in \mathcal{G}'$ and
  $r_t = \max_{G_t \in \mathcal{G}'_t} \text{diam}(G_t) \ \forall t \in I$;
- any measurable dynamic system $\text{dyn}(t, G', v)$ preserving $\sim_{\text{DUT}}$;
- any norm $\| \cdot \|$ on $\mathbb{R}$ and probability measure $P$ on $\mathcal{G}$;
- $\epsilon, \lambda \in \mathbb{R}$, $\epsilon > 0$, $\lambda \in (0, 1)$.

There exists an MP-DGNN s.t the function $\psi$ realized by the MP-DGNN, computed after $r_t + 1$ steps satisfies:

$$P \left( \| \text{dyn}(t, G', v) - \psi(G', v) \| \leq \epsilon \right) \geq 1 - \lambda.$$
Weisfeiler-Lehman goes dynamic

Conclusion

---

\(^2\) Beddar-Wiesing, D’Inverno, Graziani, Lachi, Moallem-Oureh, Scarselli, Thomas: *Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs*, arxiv preprint
Weisfeiler-Lehman goes dynamic

Conclusion

- There exist SGNNs and MP-DGNNs to approximate any measurable function on attributed and dynamic graphs to any precision and probability.
- The proof is based on attributed and dynamic WL- and UT-equivalence.

---

Beddar-Wiesing, D’Inverno, Graziani, Lachi, Moallemi-Oureh, Scarselli, Thomas: Weisfeiler-Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs, arxiv preprint
On the Extension of the Weisfeiler-Lehman Hierarchy by WL Tests for Arbitrary Graphs

Extension of the WL Hierarchy

Higher dimensional WL test
Extension of the WL Hierarchy

Higher dimensional WL test
Extension of the WL Hierarchy

Higher dimensional WL test
Extension of the WL Hierarchy

Higher dimensional WL test
Extension of the WL Hierarchy

Higher dimensional WL test

Extensions to $k$-AWL/DWL are analogously.
The WL Hierarchy

$1\text{-WL} = 2\text{-WL} \subsetneq 3\text{-WL} \subsetneq \ldots \subsetneq k - \text{WL} \subsetneq \ldots \subsetneq \text{GI}$
Extension of the WL Hierarchy

The WL Hierarchy

\[1\text{-WL} = 2\text{-WL} \subsetneq 3\text{-WL} \subsetneq \ldots \subsetneq k - \text{WL} \subsetneq \ldots \subsetneq \text{GI}\]

How do the \(k\text{-AWL}\) and \(k\text{-DWL}\) fit there?
Extension of the WL Hierarchy

Some trivial observations are:

- $1$-WL $\subsetneq$ $1$-AWL
- $\Rightarrow k$-WL $\subsetneq$ $k$-AWL
- $2$-WL $\subsetneq$ $1$-AWL
- $k$-AWL/DWL $\subseteq (k + 1)$-AWL/DWL
- $k$-AWL $=$ $k$-DWL
Extension of the WL Hierarchy

Some trivial observations are:

- 1-WL $\subsetneq$ 1-AWL
- $\Rightarrow$ $k$-WL $\subsetneq$ $k$-AWL
- 2-WL $\subsetneq$ 1-AWL
- $k$-AWL/DWL $\subseteq$ $(k + 1)$-AWL/DWL
- $k$-AWL = $k$-DWL

Nevertheless, the hierarchy can just induce a partial order:

- 3-WL $\nsubseteq$ 1-AWL
- 3-WL $\nsubseteq$ 1-AWL
- \ldots
Extension of the WL Hierarchy

3-WL $\nsubseteq$ 1-AWL

\[ \begin{array}{c}
\quad 2 \\
\downarrow \\
\quad 2 \\
\quad 2 \\
\quad 4 \\
\quad 2 \\
\quad 2 \\
\quad 3 \\
\end{array} \quad \Leftrightarrow \quad \begin{array}{c}
\quad 2 \\
\downarrow \\
\quad 2 \\
\quad 2 \\
\quad 4 \\
\quad 2 \\
\quad 2 \\
\quad 3 \\
\end{array} \]
Extension of the WL Hierarchy

3-WL $\not\subseteq$ 1-AWL

3-WL $\not\subseteq$ 1-AWL
Extension of the WL Hierarchy

- Extended WL Hierarchy induces a lattice.
Extended WL Hierarchy induces a lattice.

Def.: A lattice is $(L, \land, \lor)$, with set $L$ and associative and commutative operations $\land, \lor$ fulfilling the absorption and the idempotent laws.
Extension of the WL Hierarchy

- Extended WL Hierarchy induces a **lattice**.
- Def.: A lattice is \((L, \wedge, \vee)\), with set \(L\) and associative and commutative operations \(\wedge, \vee\) fulfilling the absorption and the idempotent laws.
- Lattice is complete, infinite, bounded, distributive and modular.
Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
Extension of the WL Hierarchy

Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
  - How big is the difference $|\mathcal{P}_B| - |\mathcal{P}_A|$ of the partitions $\mathcal{P}_A, \mathcal{P}_B$ if $A \leq B$?
Extension of the WL Hierarchy

Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
  - How big is the difference $|P_B| - |P_A|$ of the partitions $P_A, P_B$ if $A \leq B$?
  - Is it possible for two graphs to find the minimal WL test capable of distinguishing the graphs?
Extension of the WL Hierarchy

Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
  - How big is the difference $|\mathcal{P}_B| - |\mathcal{P}_A|$ of the partitions $\mathcal{P}_A, \mathcal{P}_B$ if $A \leq B$?
  - Is it possible for two graphs to find the minimal WL test capable of distinguishing the graphs?
  - What are minimal requirements to a subset of WL tests such that it remains a lattice, or that we obtain a semilattice?
Extension of the WL Hierarchy

**Why are these results so great?**
- We could use lattice theory to solve open questions as e.g.:
  - How big is the *difference* $|\mathcal{P}_B| - |\mathcal{P}_A|$ of the partitions $\mathcal{P}_A, \mathcal{P}_B$ if $A \leq B$?
  - Is it possible for two graphs to find the *minimal WL test* capable of distinguishing the graphs?
  - What are *minimal requirements* to a subset of WL tests such that it remains a *lattice*, or that we obtain a *semilattice*?

**Further future work**
- The $k$-AWL/DWL extensions earlier are very simple, but mirror the GNN architecture.
Extension of the WL Hierarchy

Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
  - How big is the difference $|\mathcal{P}_B| - |\mathcal{P}_A|$ of the partitions $\mathcal{P}_A, \mathcal{P}_B$ if $A \leq B$?
  - Is it possible for two graphs to find the minimal WL test capable of distinguishing the graphs?
  - What are minimal requirements to a subset of WL tests such that it remains a lattice, or that we obtain a semilattice?

Further future work

- The $k$-AWL/DWL extensions earlier are very simple, but mirror the GNN architecture.
- There are more powerful extensions (without this property).
Extension of the WL Hierarchy

Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
  - How big is the difference $|\mathcal{P}_B| - |\mathcal{P}_A|$ of the partitions $\mathcal{P}_A, \mathcal{P}_B$ if $A \leq B$?
  - Is it possible for two graphs to find the minimal WL test capable of distinguishing the graphs?
  - What are minimal requirements to a subset of WL tests such that it remains a lattice, or that we obtain a semilattice?

Further future work

- The $k$-AWL/DWL extensions earlier are very simple, but mirror the GNN architecture.
- There are more powerful extensions (without this property).
- → How would these change the WL lattice?
Extension of the WL Hierarchy

Why are these results so great?

- We could use lattice theory to solve open questions as e.g.:
  - How big is the difference $|P_B| - |P_A|$ of the partitions $P_A$, $P_B$ if $A \leq B$?
  - Is it possible for two graphs to find the minimal WL test capable of distinguishing the graphs?
  - What are minimal requirements to a subset of WL tests such that it remains a lattice, or that we obtain a semilattice?

Further future work

- The $k$-AWL/DWL extensions earlier are very simple, but mirror the GNN architecture.
- There are more powerful extensions (without this property).
- How would these change the WL lattice?

Since this is future work, feel free to share your expertise!
Graph Neural Networks for Power Grids
The Importance of the Power Grid

- Reliability and safety of the power grid is essential
The Importance of the Power Grid

- Reliability and safety of the power grid is essential
- The power grid is a complex system, which has to adapt to changing conditions
The Importance of the Power Grid

- Reliability and safety of the power grid is essential
- The power grid is a complex system, which has to adapt to changing conditions
- Fluctuations caused by renewable energies require a high flexiblility
The Importance of the Power Grid

- Reliability and safety of the power grid is essential
- The power grid is a complex system, which has to adapt to changing conditions
- Fluctuations caused by renewable energies require a high flexiblility
- Efficient power grid Operation is required for a successful decarbonization
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispacth,
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch, **topological operations**
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch, **topological operations**
- Topology changes are typically low cost actions
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch, **topological operations**
- Topology changes are typically low cost actions
- Simulating every action is not feasible
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch, topological operations
- Topology changes are typically low cost actions
- Simulating every action is not feasible
- Topology changes are underexploited options
Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch, **topological operations**
- Topology changes are typically low cost actions
- Simulating every action is not feasible
- Topology changes are underexploited options

⇒ Deep Learning Models
GNNs for Electricity Networks
GNNs for Electricity Networks

- The power grid has an inherent graph structure
GNNs for Electricity Networks

- The power grid has an inherent graph structure
- Its components are strongly correlated
GNNs for Electricity Networks

- The power grid has an inherent graph structure
- Its components are strongly correlated
- GNNs can leverage the power grid’s topology to generate a graph output
Graph Neural Networks for Power Grids

GNNs for Electricity Networks

- The power grid has an inherent graph structure
- Its components are strongly correlated
- GNNs can leverage the power grid’s topology to generate a graph output

Goal: Design a GNN to predict a suitable topology
Graph Neural Networks for Power Grids

Approach

- Input: power grid at a specific time stamp
- Construct Graph
- Apply GNN

- Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction
Graph Neural Networks for Power Grids

**Approach**

- **Input:** power grid at a specific time stamp
- **Construct Graph**
- **Apply GNN**

- **Output:** Encoded Graph indicating the splitting candidates
- **Change topology according to prediction**
Graph Neural Networks for Power Grids

**Approach**

- **Input:** power grid at a specific time stamp
- **Construct Graph**
- **Apply GNN**

**Output:** Encoded Graph indicating the splitting candidates

- **Change topology according to prediction**
Graph Neural Networks for Power Grids

Approach

- Input: power grid at a specific time stamp
- Construct Graph
- Apply GNN

Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction
Graph Neural Networks for Power Grids

**Approach**

- Input: power grid at a specific time stamp
- Construct Graph
- Apply GNN

- Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction
Graph Neural Networks for Power Grids

**Approach**

- **Input:** power grid at a specific time stamp
- **Construct Graph**
- **Apply GNN**

**Output:** Encoded Graph indicating the splitting candidates

- **Change topology according to prediction**

→ **Node Classification Task to identify the candidates for node splitting**
Further Considerations

- Hardly any approaches for this specific use case
Further Considerations

- Hardly any approaches for this specific use case
- Apply dedicated GNN architecture
Further Considerations

- Hardly any approaches for this specific use case
- Apply dedicated GNN architecture
- Include historical data
Further Considerations

- Hardly any approaches for this specific use case
- Apply dedicated GNN architecture
- Include historical data
  - Time Series Embedding
Further Considerations

- Hardly any approaches for this specific use case
- Apply dedicated GNN architecture
- Include historical data
  - Time Series Embedding
  - Dynamic GNN
Ultimate Goal

- Combine the GNN Approach with a Reinforcement Learning Algorithm
Learning to Run a Power Network

---

3 Marot, Antoine and Donnot, Benjamin and Romero, Camilo and Donon, Balthazar and Lerousseau, Marvin and Veyrin-Forrer, Luca and Guyon, Isabelle: Learning to run a power network challenge for training topology controllers, Electric Power Systems Research vol. 189, Elsevier
Graph Neural Networks for Power Grids

Learning to Run a Power Network

- NeurIPS Challenge "L2RPN"\(^3\)

---

\(^3\) Marot, Antoine and Donnot, Benjamin and Romero, Camilo and Donon, Balthazar and Lerousseau, Marvin and Veyrin-Forrer, Luca and Guyon, Isabelle: *Learning to run a power network challenge for training topology controllers*, Electric Power Systems Research vol. 189, Elsevier
Learning to Run a Power Network

- NeurIPS Challenge "L2RPN"\(^3\)
- Hardly any Agents using GNNs

---

\(^3\) Marot, Antoine and Donnot, Benjamin and Romero, Camilo and Donon, Balthazar and Lerousseau, Marvin and Veyrin-Forrer, Luca and Guyon, Isabelle: *Learning to run a power network challenge for training topology controllers*, Electric Power Systems Research vol. 189, Elsevier
Learning to Run a Power Network

- NeurIPS Challenge "L2RPN"\textsuperscript{3}
- Hardly any Agents using GNNs
- Influence of GNNs has not been fully investigated

\textsuperscript{3} Marot, Antoine and Donnot, Benjamin and Romero, Camilo and Donon, Balthazar and Lerousseau, Marvin and Veyrin-Forrer, Luca and Guyon, Isabelle: \textit{Learning to run a power network challenge for training topology controllers}, Electric Power Systems Research vol. 189, Elsevier
Graph Neural Networks for Power Grids

Learning to Run a Power Network

- NeurIPS Challenge "L2RPN"\(^3\)
- Hardly any Agents using GNNs
- Influence of GNNs has not been fully investigated
- GNNs for Imitation Learning as benchmark

---

\(^3\) Marot, Antoine and Donnot, Benjamin and Romero, Camilo and Donon, Balthazar and Lerousseau, Marvin and Veyrin-Forrer, Luca and Guyon, Isabelle: *Learning to run a power network challenge for training topology controllers*, Electric Power Systems Research vol. 189, Elsevier
Graph Neural Networks for Power Grids

GNN Pipeline with the Gri2Op\(^4\) Virtual Power Grid

---

4. B. Donnot, Grid2op- A testbed platform to model sequential decision making in power systems. 
https://GitHub.com/rte-france/grid2op
Ongoing Research
Local Activity Encoding for Dynamic Graph Pooling in Structure Dynamic Graphs
Continuous-Time Generative GNN for Attributed Dynamic Graphs
FDGNN: Fully Dynamic GNN
Local Activity Encoding for Dynamic Graph Pooling in Structure Dynamic Graphs

- graph compression algorithm for processing structural dynamic graphs
- includes local activity encoding with subsequent pooling
- generates important graph sequence of equal sizes in $\mathcal{O}(T)$
Let $G = (g_{t_0}, E)$ be a dynamic graph in continuous-time. The approach is determined by:

1. Discretization of $G$
2. Embedding via vGAE
3. Interpret timestamps as another embedding space scaling axis and fit Gaussian regression functions

- emb. coord. of node $n$ at time $t_i$
- emb. forecast coord. of node $n$ at time $t_{i+1}$

→ polynomial regression function $f_n(t)$
FDGNN: Fully Dynamic Graph Neural Network

- node/edge activity: existence of nodes/edges over time
- attribute embeddings: latent vector representations of attributes
- self-attention: self-propagation gate
- neighborhood-attention: neighborhood-propagation gate

- node/edge embeddings
- exogenous drive (temporal delay)
- intensity functions: occurrence of events (node/edge additions/deletions/attribute changes)

FDGNN: Fully Dynamic Graph Neural Network

- **node/edge activity**: existence of nodes/edges over time
- **attribute embeddings**: latent vector representations of attributes
- **self-attention**: self-propagation gate
- **neighborhood-attention**: neighborhood-propagation gate
- **node/edge embeddings**: exogeneous drive (temporal delay)
- **intensity functions**: occurrence of events (node/edge additions/deletions/attribute changes)

---

FDGNN: Fully Dynamic Graph Neural Network

- **node/edge activity**: existence of nodes/edges over time
- **attribute embeddings**: latent vector representations of attributes
- **self-attention**: self-propagation gate
- **neighborhood-attention**: neighborhood-propagation gate
- **node/edge embeddings**: exogeneous drive (temporal delay)
- **intensity functions**: occurrence of events (node/edge additions/deletions/attribute changes)

---

FDGNN: Fully Dynamic Graph Neural Network

- **node/edge activity**: existence of nodes/edges over time
- **attribute embeddings**: latent vector representations of attributes
- **self-attention**: self-propagation gate
- **neighborhood-attention**: neighborhood-propagation gate

**Node/edge embeddings**
- **exogeneous drive (temporal delay)**
- **occurrence of events (node/edge additions/deletions/attribute changes)**

FDGNN: Fully Dynamic Graph Neural Network

node/edge activity
existence of nodes/edges over time

attribute embeddings
latent vector representations of attributes

self-attention
self-propagation gate

neighborhood-attention
neighborhood-propagation gate

node/edge embeddings
exogenous drive (temporal delay)

intensity functions
occurrence of events (node/edge additions/deletions/attribute changes)

FDGNN: Fully Dynamic Graph Neural Network

- **Node/Edge Activity**: Existence of nodes/edges over time
- **Attribute Embeddings**: Latent vector representations of attributes
- **Self-Attention**: Self-propagation gate
- **Neighborhood Attention**: Neighborhood-propagation gate
- **Exogeneous Drive (Temporal Delay)**: Occurrence of events (node/edge additions/deletions/attribute changes)
- **Intensity Functions**
Thank you for your attention!
Questions?

Silvia Beddar-Wiesing
s.beddarwiesing@uni-kassel.de

Alice Moallem-Oureh
amoallem@uni-kassel.de

Clara Holzhüter
clara.holzhueter@uni-kassel.de
clara.juliane.holzhueter@iee.fraunhofer.de

gain-group.de