Hot Topics in Graph Neural Networks

Graphs in Artificial Intelligence and Neural Networks

Josephine Thomas, Silvia Beddar-Wiesing, Clara Holzhüter, Alice Moallemy-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 souvereignty
- 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

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

Content

Graph Neural Networks for different Graph Types: A Survey

Josephine M. Thomas*, Alice Moallemy-Oureh*, Silvia Beddar-Wiesing*, Clara Holzhüter*: *Graph Neural Networks Designed for Different Graph Types: A Survey*, `https://arxiv.org/abs/2204.03080`

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?

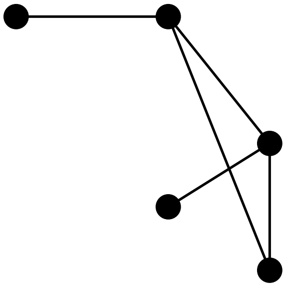
- GNNs extend Neural Networks to work on graphs
- The architecture of GNNs can be different depending on the properties of a graph
- Graph properties yield graph types → Which graph types are there?

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 $\rightarrow$ Which graph types are there?
- Which graph types can be handled by GNN models?

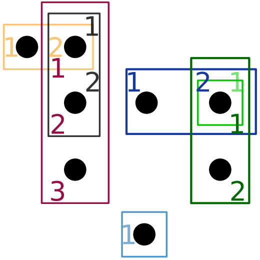
GNNs for different Graph Types: The graph types

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



GNNs for different Graph Types: The graph types

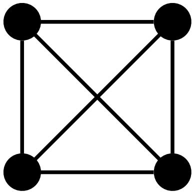
- static undirected graph
- static structural properties
- semantic graph properties
- dynamic structural properties



directed hypergraph

GNNs for different Graph Types: The graph types

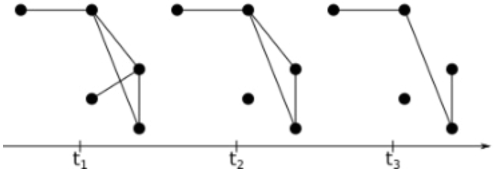
- static undirected graph
- static structural properties
- semantic graph properties
- dynamic structural properties



complete graph

GNNs for different Graph Types: The graph types

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

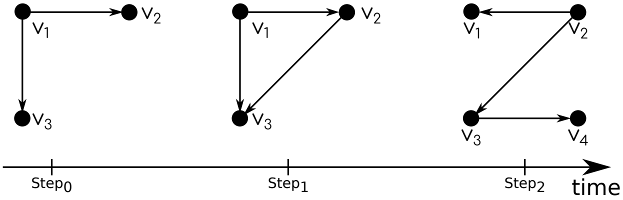


strictly shrinking graph

GNNs for different Graph Types: The representation of dynamic graphs

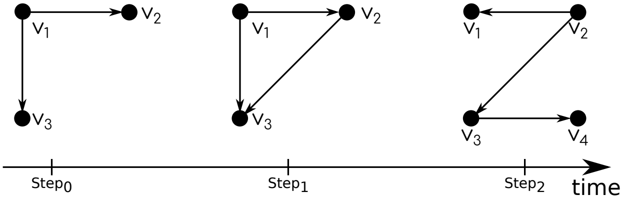
GNNs for different Graph Types: The representation of dynamic graphs

- discrete-time dynamic

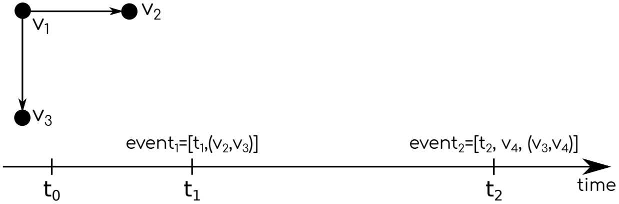


GNNs for different Graph Types: The representation of dynamic graphs

- discrete-time dynamic



- continuous-time dynamic



GNNs for different Graph Types: Where are the gaps?

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

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

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

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
- 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

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
- 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

GNNs for different Graph Types: Where are the gaps?

- 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 handeled

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 handeled
  - 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 handeled
  - 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 handeled
  - 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 handeled
  - 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 handeled
  - 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 handeled
  - 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 handeled
  - unconnected graphs
  - acyclic graphs
  - r-regular graphs
- Many gaps in models for **hypergraphs in continuous-time**
  - Node/edge-attributes
  - Node/edge-heterogenity
  - Multiple nodes/eges

The Modeling Power of different Graph Types

Josephine M. Thomas*, Silvia Beddar-Wiesing*, Alice Moallemy-Oureh*, Rüdiger Nather: *Graph type expressivity and transformations*, `https://arxiv.org/abs/2109.10708`

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

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

The Modeling Power of different Graph Types

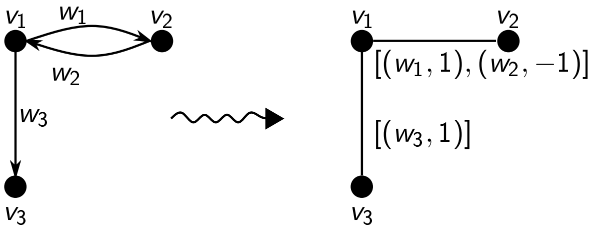
How do we assess the ability of different graph types to represent information?

- 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 $\mathcal{G}_2$ is **at least as expressive** as a graph type $\mathcal{G}_1$, if and only if $\mathcal{G}_2$ encodes at least as many graph properties as $\mathcal{G}_1$ denoted as $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$. In case both types encode the same graph properties it is denoted as $\mathcal{G}_1 \approx \mathcal{G}_2$.

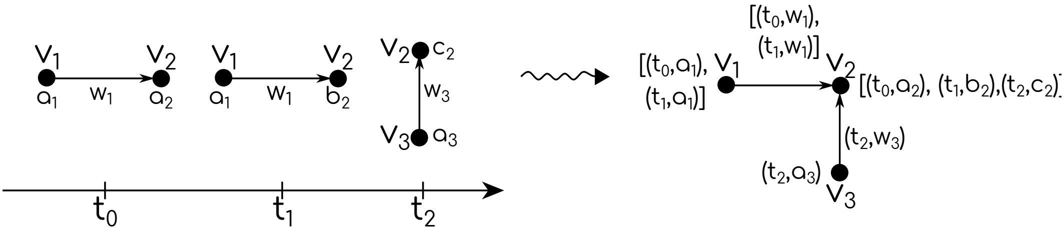
The Modeling Power of different Graph Types: Expres. relation examples



$$\mathcal{G}_{directed} \preccurlyeq \mathcal{G}_{undirected}$$

Transformation directed to undirected graph:
Storing the directions and multiple attributes in the new attributes.

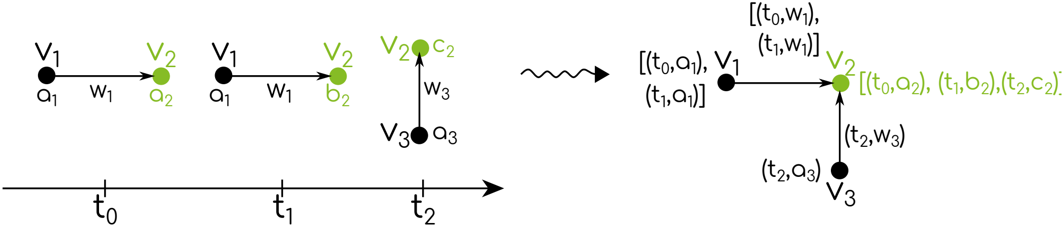
The Modeling Power of different Graph Types: Expres. relation examples



$\mathcal{G}_{dynamic} \preccurlyeq \mathcal{G}_{static}$

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: Expres. relation examples



$\mathcal{G}_{dynamic} \preccurlyeq \mathcal{G}_{static}$

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)**.

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 are equally expressive.

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 are equally expressive.

$\rightarrow$ We can transform graph data to be able to use an arbitrary GNN.

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 are equally expressive.

$\rightarrow$ We can transform graph data to be able to use an arbitrary GNN.
$\rightarrow$ We are free to choose a graph type that models our problem best.

UNIVERSITÀ DI SIENA 1240

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

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.

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

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.

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.

$\rightarrow$ static node-attributed graphs only!

Contributions

[1] Beddar-Wiesing, D'Inverno, Graziani, Lachi, Moallemy-Oureh, Scarselli, Thomas: *Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs*, arxiv preprint

Contributions

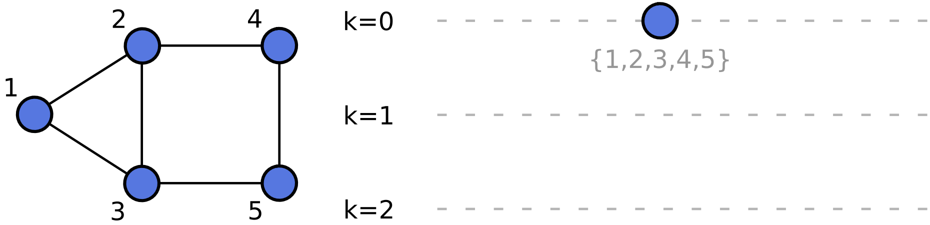
- **Extension** of WL-Tests and unfolding trees to (edge-)attributes and dynamics
- **Proof of Extended Approximation Theorems:** GNNs can approximate to any precision and probability any measurable function on attributed and dynamic graphs

[1] Beddar-Wiesing, D'Inverno, Graziani, Lachi, Moallemy-Oureh, Scarselli, Thomas: *Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs*, arxiv preprint

Recap: WL-Test and Unfolding Trees

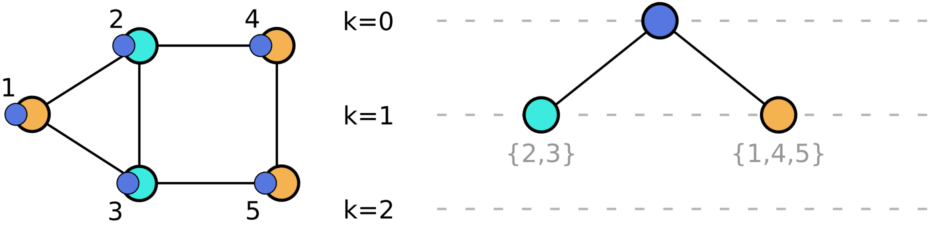
Weisfeiler-Lehman goes dynamic

Recap: WL-Test and Unfolding Trees



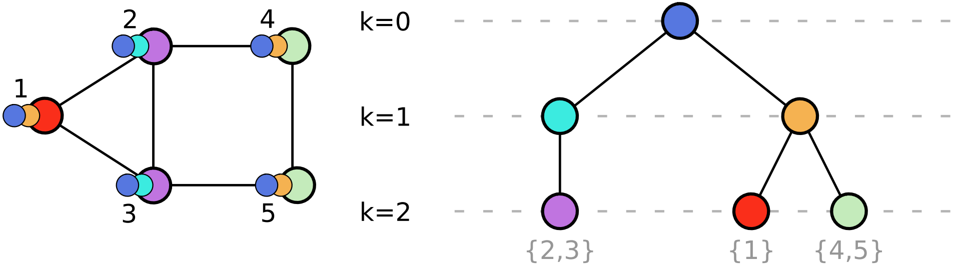
Weisfeiler-Lehman goes dynamic

Recap: WL-Test and Unfolding Trees



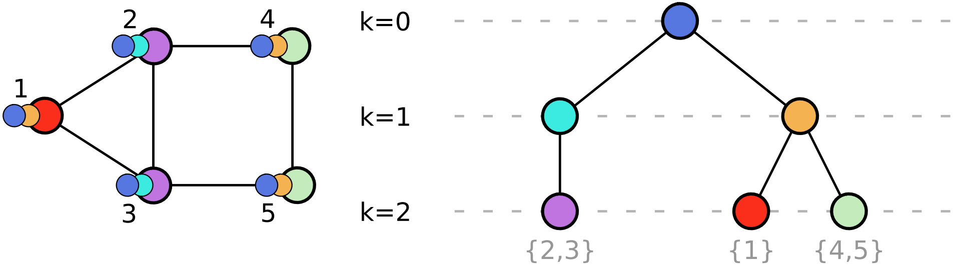
Weisfeiler-Lehman goes dynamic

Recap: WL-Test and Unfolding Trees



Weisfeiler-Lehman goes dynamic

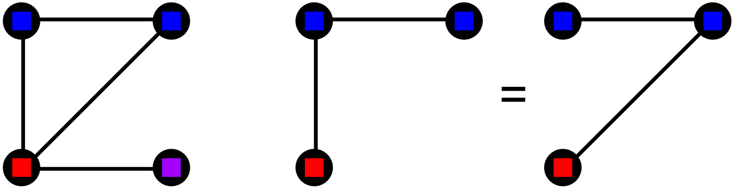
Recap: WL-Test and Unfolding Trees



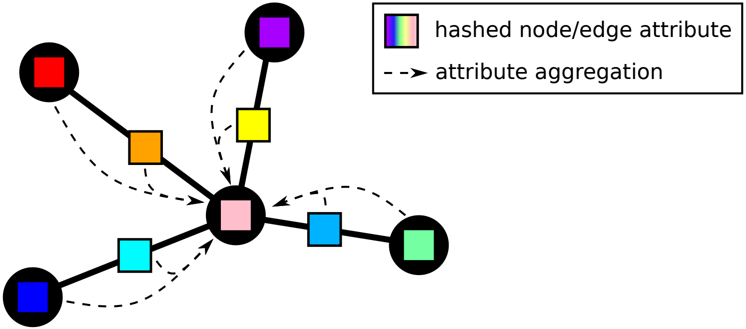
(Thank you Nils Kriege for the wonderful illustration!)

Unfolding Trees

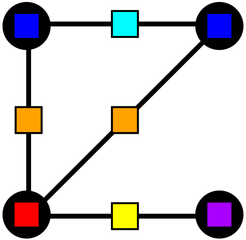
Unfolding Trees of both blue nodes



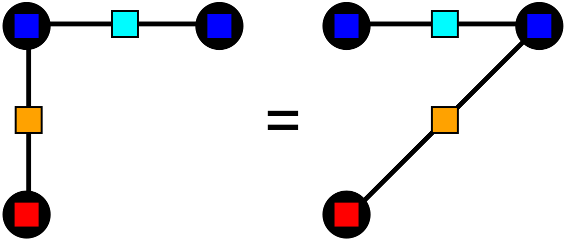
WL Coloring for Attributed Graphs



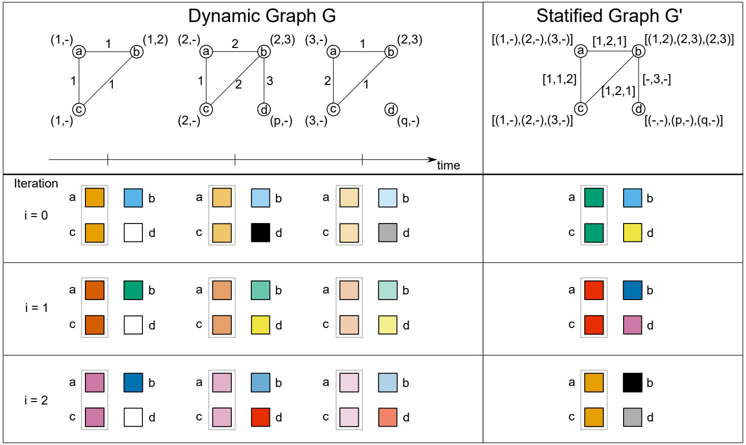
Unfolding Trees for Attributed Graphs



Unfolding Trees of both blue nodes



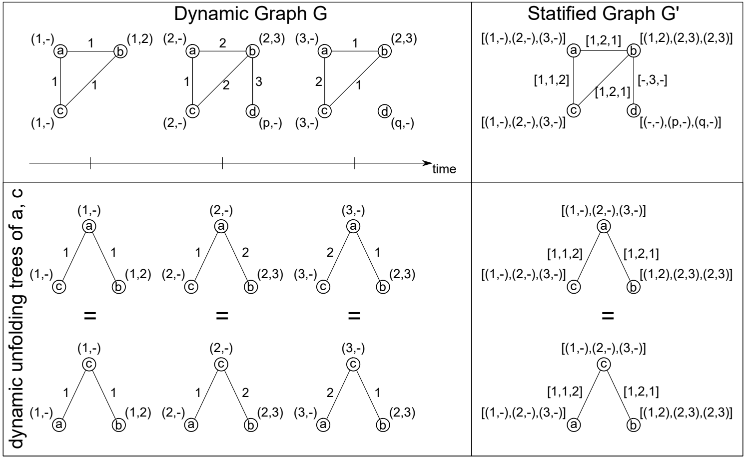
WL Coloring for Dynamic Graphs



Weisfeiler-Lehman goes dynamic

Extension of WL-Test and Unfolding Trees

Unfolding Trees for Dynamic Graphs



Proposition

For all nodes u, v holds:

1. in the attributed case:

$$u \sim_{AWL} v \Leftrightarrow u \sim_{AUT} v.$$

2. in the dynamic case:

$$u \sim_{DWL} v \Leftrightarrow u \sim_{DUT} v.$$

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

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

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

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

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

$$h_v^k(t) = \text{COMBINE}_t^{(k)}\left(\underbrace{h_v^{k-1}(t)}_{\text{history}}, \underbrace{\text{AGGREGATE}_t^k\left(\{h_u^{k-1}(t)\}_{u\in\mathcal{N}_t(v)}, \{\omega_{\{u,v\}}(t)\}_{u\in\mathcal{N}_t(v)}\right)}_{\text{temporal neighborhood aggregation}}\right)$$

Weisfeiler-Lehman goes dynamic

Universal Approximation of SGNN and MP-DGNN

Universal Approximation of SGNN and MP-DGNN

For

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

For

- Domain of SAUHGs $\mathcal{G}$ and $r = \max_{g \in \mathcal{G}} diam(G)$;
- any measurable function f preserving $\sim_{AUT}$;
- 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 φ 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 $r = \max_{g \in \mathcal{G}} diam(G)$;
- any measurable function f preserving $\sim_{AUT}$;
- 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 φ 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 $r_t = \max_{G_t \in \mathcal{G}'} diam(G_t)\ \forall t \in I$;
- any measurable dynamic system $dyn(t, G', v)$ preserving $\sim_{DUT}$;
- 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 ψ realized by the MP-DGNN, computed after $r_t + 1$ steps satisfies:

$$P\left(\|dyn(t, G', v) - \psi(G', v)\| \leq \epsilon\right) \geq 1 - \lambda.$$

Conclusion

[2] Beddar-Wiesing, D'Inverno, Graziani, Lachi, Moallemy-Oureh, Scarselli, Thomas: *Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs*, arxiv preprint

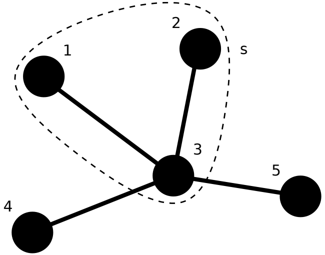
- There **exist** SGNNs and MP-DGNNs to **approximate any measurable function** on attributed and dynamic graphs to any precision and probability.
- The **proof** is based on attributed and dynamic WL- and UT- equivalence.

[2] Beddar-Wiesing, D'Inverno, Graziani, Lachi, Moallemy-Oureh, Scarselli, Thomas: *Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs*, arxiv preprint

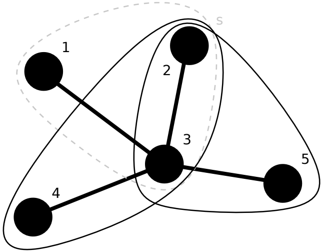
On the Extension of the Weisfeiler-Lehman Hierarchy by WL Tests for Arbitrary Graphs

S. Beddar-Wiesing, G.A. D'Inverno, C. Graziani, V. Lachi, A. Moallemy-Oureh, F: Scarselli *On the Extension of the Weisfeiler-Lehman Hierarchy by WL Tests for Arbitrary Graphs*, 18th International Workshop On Mining and Learning with Graphs, 2022,
`https://openreview.net/forum?id=Qt6GrgDz2y5`

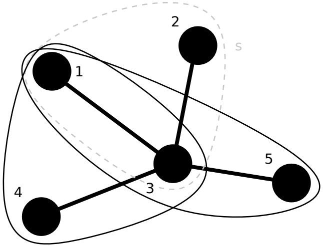
Higher dimensional WL test



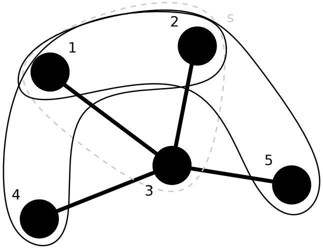
Higher dimensional WL test



Higher dimensional WL test

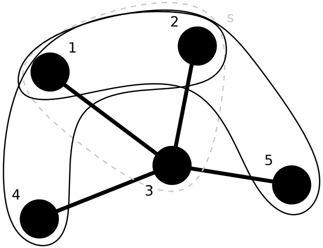


Higher dimensional WL test



Extension of the WL Hierarchy

Higher dimensional WL test



Extensions to k-AWL/DWL are analogously.

Extension of the WL Hierarchy

The WL Hierachy

$$\textbf{1-WL} = \textbf{2-WL} \subsetneq \textbf{3-WL} \subsetneq \ldots \subsetneq k - \textbf{WL} \subsetneq \ldots \subsetneq \textbf{GI}$$

Extension of the WL Hierarchy

The WL Hierachy

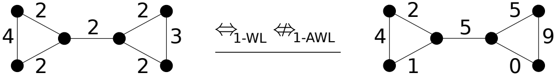
$$\textbf{1-WL} = \textbf{2-WL} \subsetneq \textbf{3-WL} \subsetneq \ldots \subsetneq k - \textbf{WL} \subsetneq \ldots \subsetneq \textbf{GI}$$

How do the *k*-AWL and *k*-DWL fit there?

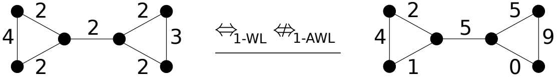
Extension of the WL Hierarchy

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



Some trivial observations are:

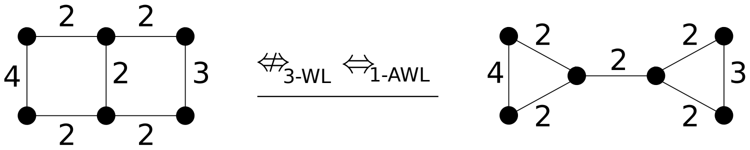


- $1\text{-WL} \subsetneq 1\text{-AWL}$
- $\Rightarrow k\text{-WL} \subsetneq k\text{-AWL}$
- $2\text{-WL} \subsetneq 1\text{-AWL}$
- $k\text{-AWL/DWL} \subseteq (k+1)\text{-AWL/DWL}$
- $k\text{-AWL} = k\text{-DWL}$

Nevertheless, the hierarchy can just induce a partial order:

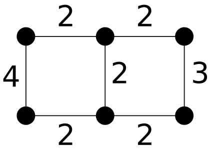
- $3\text{-WL} \not\subseteq 1\text{-AWL}$
- $3\text{-WL} \not\supseteq 1\text{-AWL}$
- ...

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

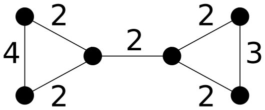


Extension of the WL Hierarchy

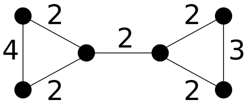
3-WL $\not\subseteq$ 1-AWL



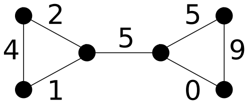
$\not\Leftrightarrow_{3\text{-WL}} \; \Leftrightarrow_{1\text{-AWL}}$



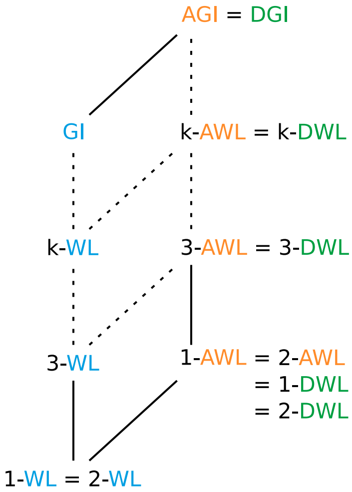
3-WL $\not\supseteq$ 1-AWL



$\Leftrightarrow_{3\text{-WL}} \; \not\Leftrightarrow_{1\text{-AWL}}$

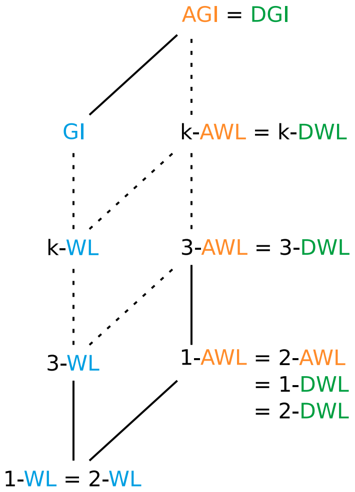


Extension of the WL Hierarchy



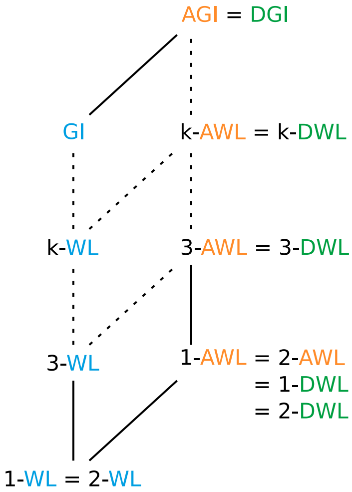
- Extended WL Hierarchy induces a **lattice.**

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.

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.:

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 $A \leq B$?

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?

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**?

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.

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).

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).
- $\longrightarrow$ How would these change the WL lattice?

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).
- $\longrightarrow$ 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

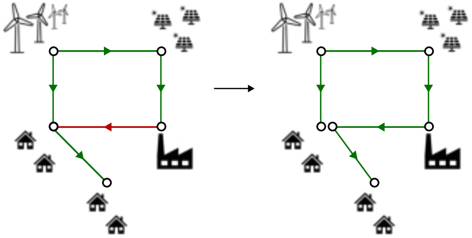
- Massive amount of regulatory actions available for the network operators

Power Grid Operation

- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch,

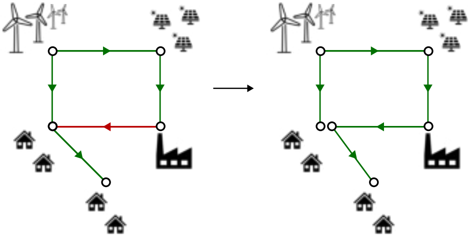
Power Grid Operation

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



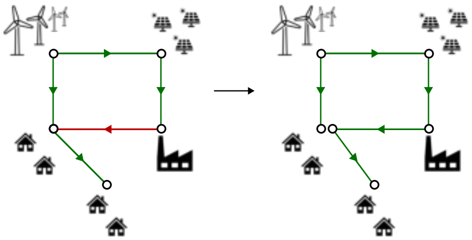
Power Grid Operation

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



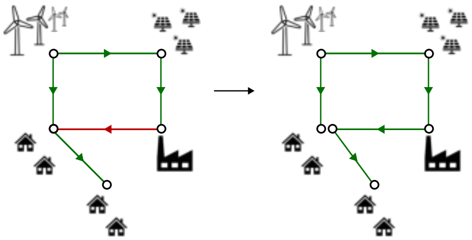
Power Grid Operation

- 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

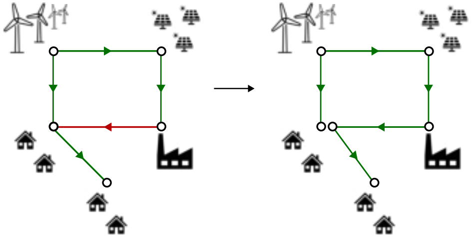
- Massive amount of regulatory actions available for the network operators
- Different actions: redispatch, **topological operations**
- Topology changes are typically low cost actions
- Simulating every action is not feasible
- Topology changes are underexploited options



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

$\Rightarrow$ 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

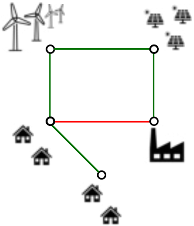
GNNs for Electricity Networks

- 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

Approach

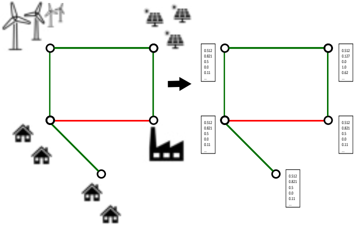
- Input: power grid at a specific time stamp
- Construct Graph
- Apply GNN
- Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction



Graph Neural Networks for Power Grids

Approach

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

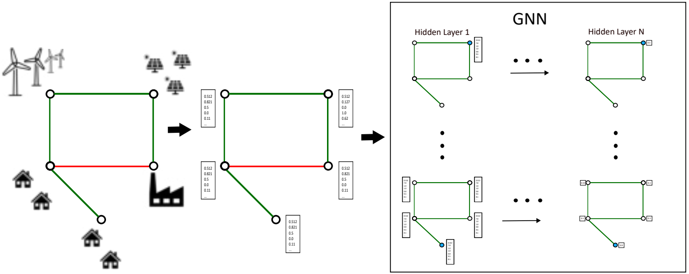


- Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction

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

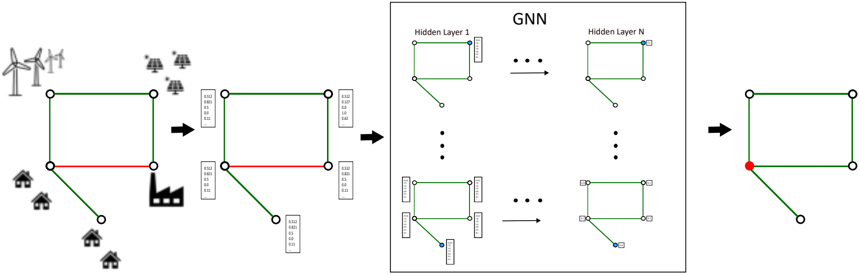


Graph Neural Networks for Power Grids

Approach

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

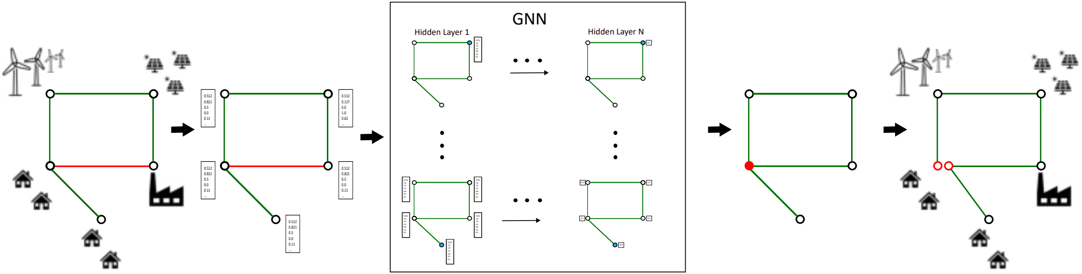
- Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction



Graph Neural Networks for Power Grids

Approach

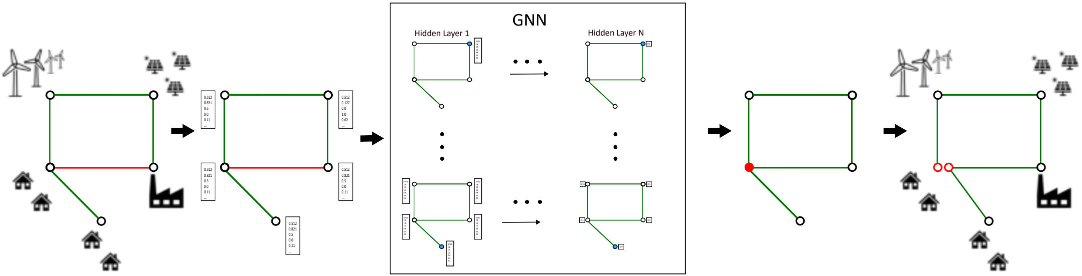
- Input: power grid at a specific time stamp
- Construct Graph
- Apply GNN
- Output: Encoded Graph indicating the splitting candidates
- Change topology according to prediction



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 Classifcation 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

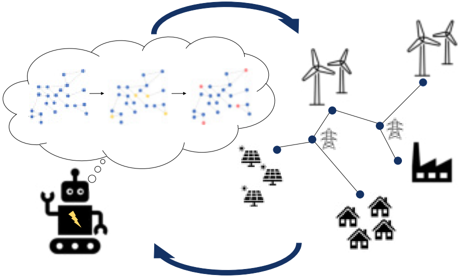
- Hardly any approaches for this specific use case
- Apply dedicated GNN architecture
- Include historical data
    - Time Series Embedding

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

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"[3]
- Hardly any Agents using GNNs
- Influence of GNNs has not been fully investigated

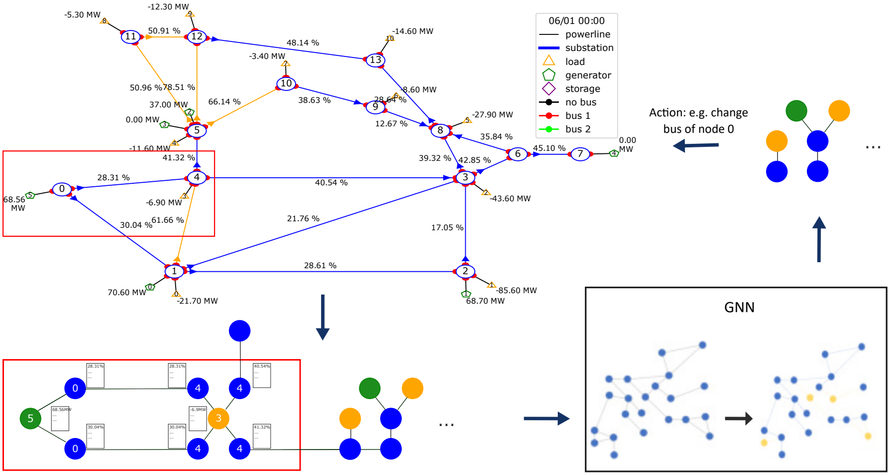
[3] Marot, Antoine and Donnot, Benjamin and Romero, Camilo and Donon, Balthazar and Lerousseau, Marvin and Veyrin-Forrer, Luca and Guyon, Isabelle: *Learning to run a power network challenge for training topology controllers*, Electric Power Systems Research vol. 189, Elsevier

Learning to Run a Power Network

- NeurIPS Challenge "L2RPN"[3]
- Hardly any Agents using GNNs
- 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

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`

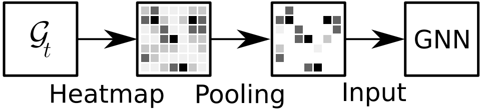
Ongoing Research

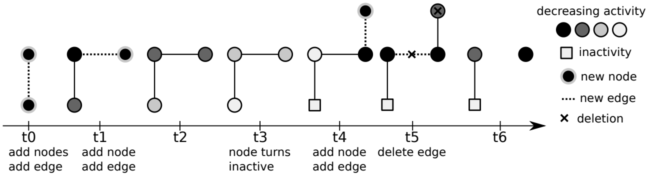
Ongoing Research

- 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[5]

- **graph compression algorithm** for processing structural dynamic graphs
- includes local **activity encoding** with subsequent **pooling**
- generates important graph sequence of **equal sizes** in $\mathcal{O}(T)$



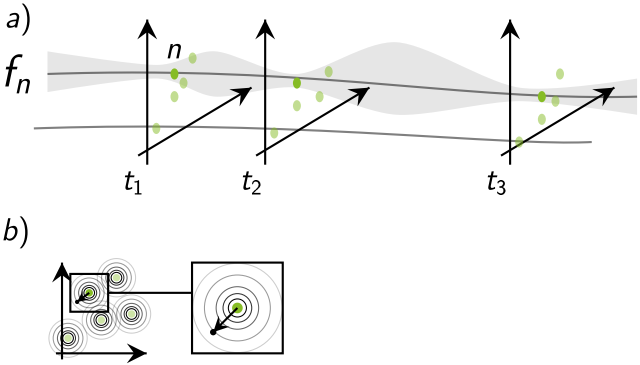


[5] Beddar-Wiesing: *Using local activity encoding for dynamic graph pooling in stuctural-dynamic graphs*, SAC '22: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing

Continuous-Time Generative GNN for Attributed Dynamic Graphs[6]

Let $G = (g_{t_0}, \mathbb{E})$ be a dynamic graph in continuous-time.
The approach is determined by:

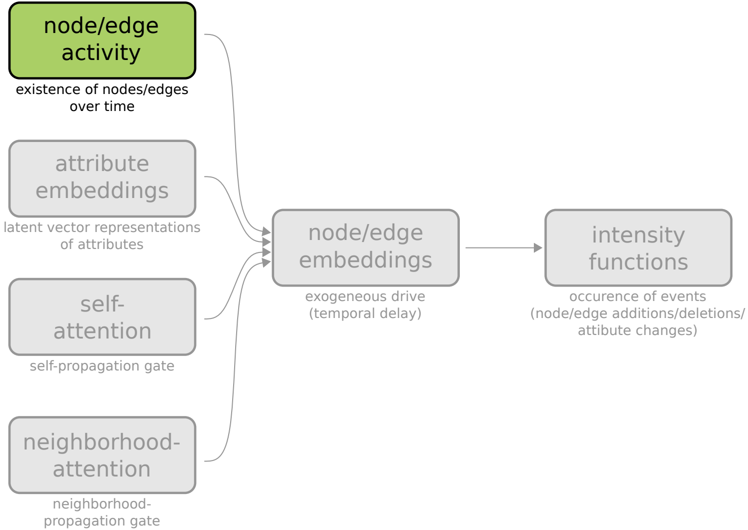
1. Discretization of G
2. Embedding via vGAE
3. Interpret timestamps as another embedding space scaling axis and fit Gaussian regression functions



- emb. coord. of node n at time t_i
- emb. forecast coord. of node n at time t_{i+1}
- → polynomial regression function $f_n(t)$

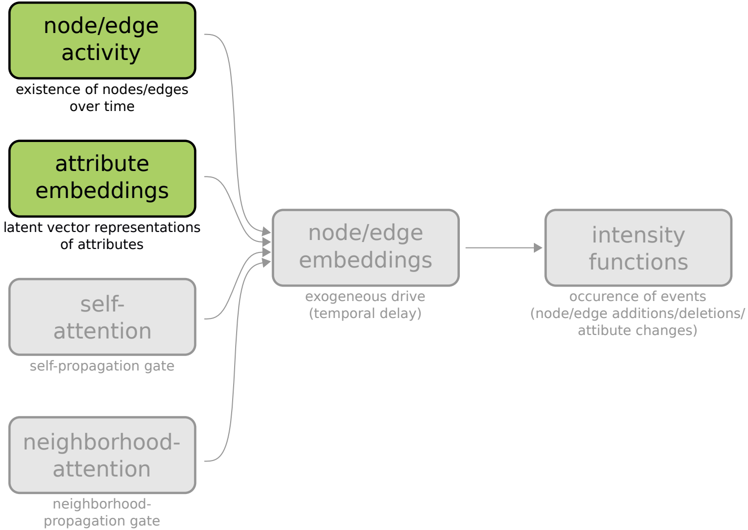
[6] Moallemy-Oureh: *Continuous-time generative graph neural network for attributed dynamic graphs*, SAC '22: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, `https://doi.org/10.1145/3477314.3508018`

FDGNN: Fully Dynamic Graph Neural Network[7]



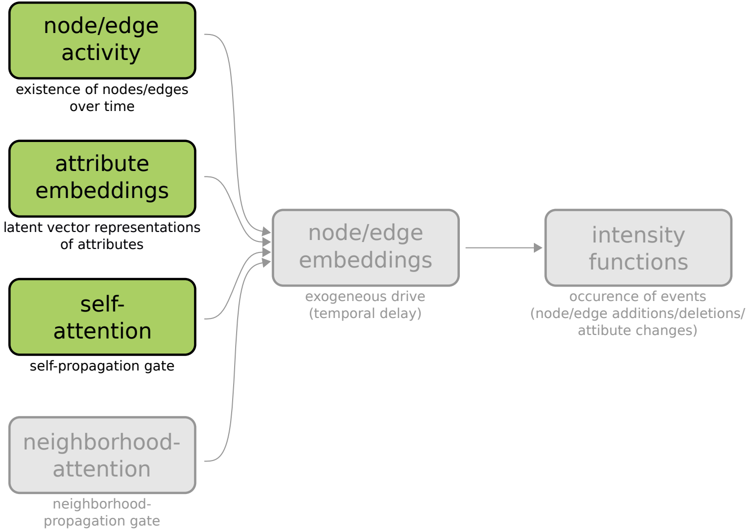
[7] Moallemy-Oureh, Beddar-Wiesing, Nather, Thomas: *FDGNN: Fully Dynamic Graph Neural Network*, arXiv:2206.03469

FDGNN: Fully Dynamic Graph Neural Network[7]



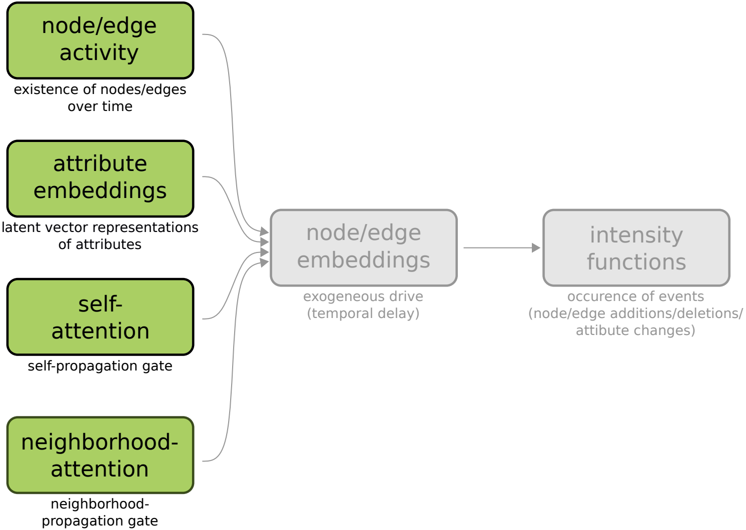
[7] Moallemy-Oureh, Beddar-Wiesing, Nather, Thomas: *FDGNN: Fully Dynamic Graph Neural Network*, arXiv:2206.03469

FDGNN: Fully Dynamic Graph Neural Network[7]



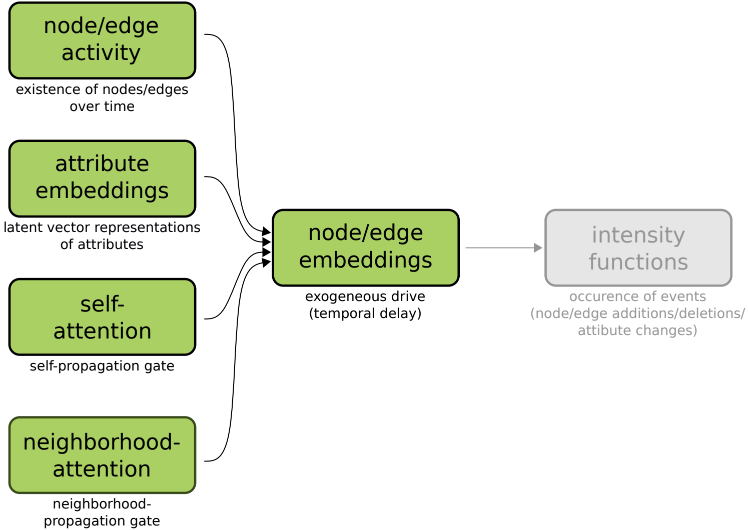
[7] Moallemy-Oureh, Beddar-Wiesing, Nather, Thomas: *FDGNN: Fully Dynamic Graph Neural Network*, arXiv:2206.03469

FDGNN: Fully Dynamic Graph Neural Network[7]



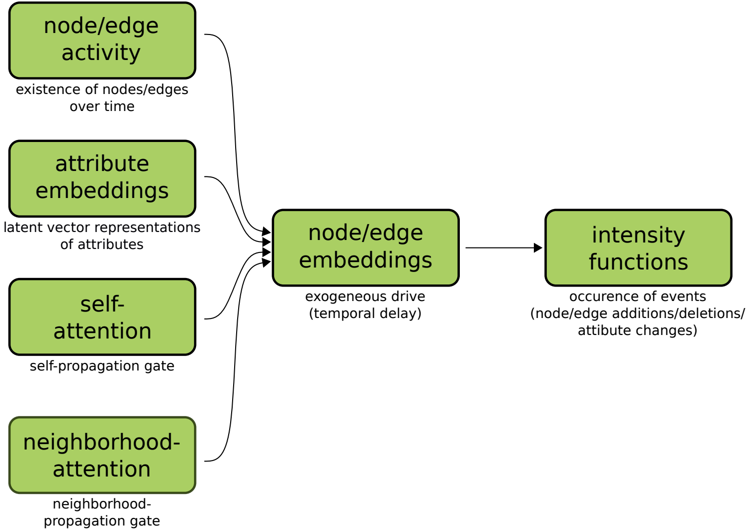
[7] Moallemy-Oureh, Beddar-Wiesing, Nather, Thomas: *FDGNN: Fully Dynamic Graph Neural Network*, arXiv:2206.03469

FDGNN: Fully Dynamic Graph Neural Network[7]



[7] Moallemy-Oureh, Beddar-Wiesing, Nather, Thomas: *FDGNN: Fully Dynamic Graph Neural Network*, arXiv:2206.03469

FDGNN: Fully Dynamic Graph Neural Network[7]



[7] Moallemy-Oureh, Beddar-Wiesing, Nather, Thomas: *FDGNN: Fully Dynamic Graph Neural Network*, arXiv:2206.03469

Contact

Thank you for your attention!
Questions?

GAIN
`gain@uni-kassel.de`

Josephine Thomas
`jthomas@uni-kassel.de`

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

Alice Moallemy-Oureh
`amoallemy@uni-kassel.de`

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

gain-group.de