Graph Neural Networks as Application of Distributed Algorithms

Roger Wattenhofer


DC

Concurrency
& Consensus

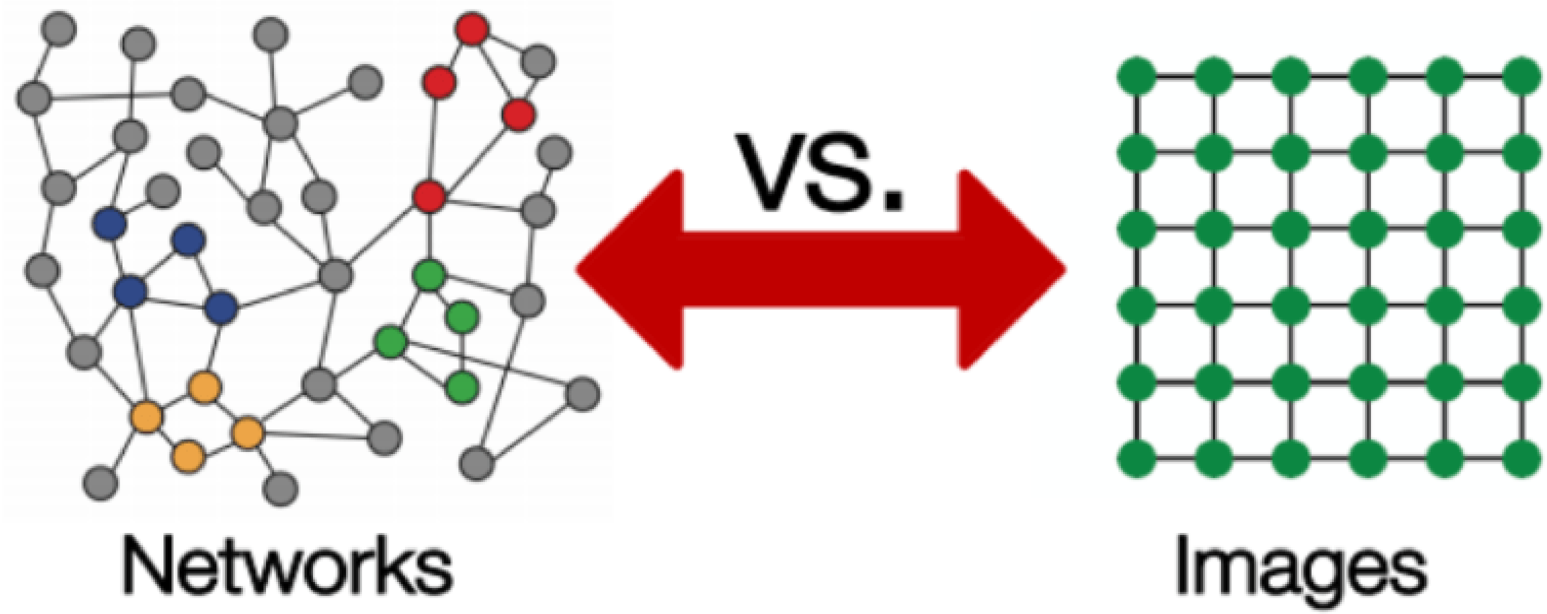
Applications?
Tons!

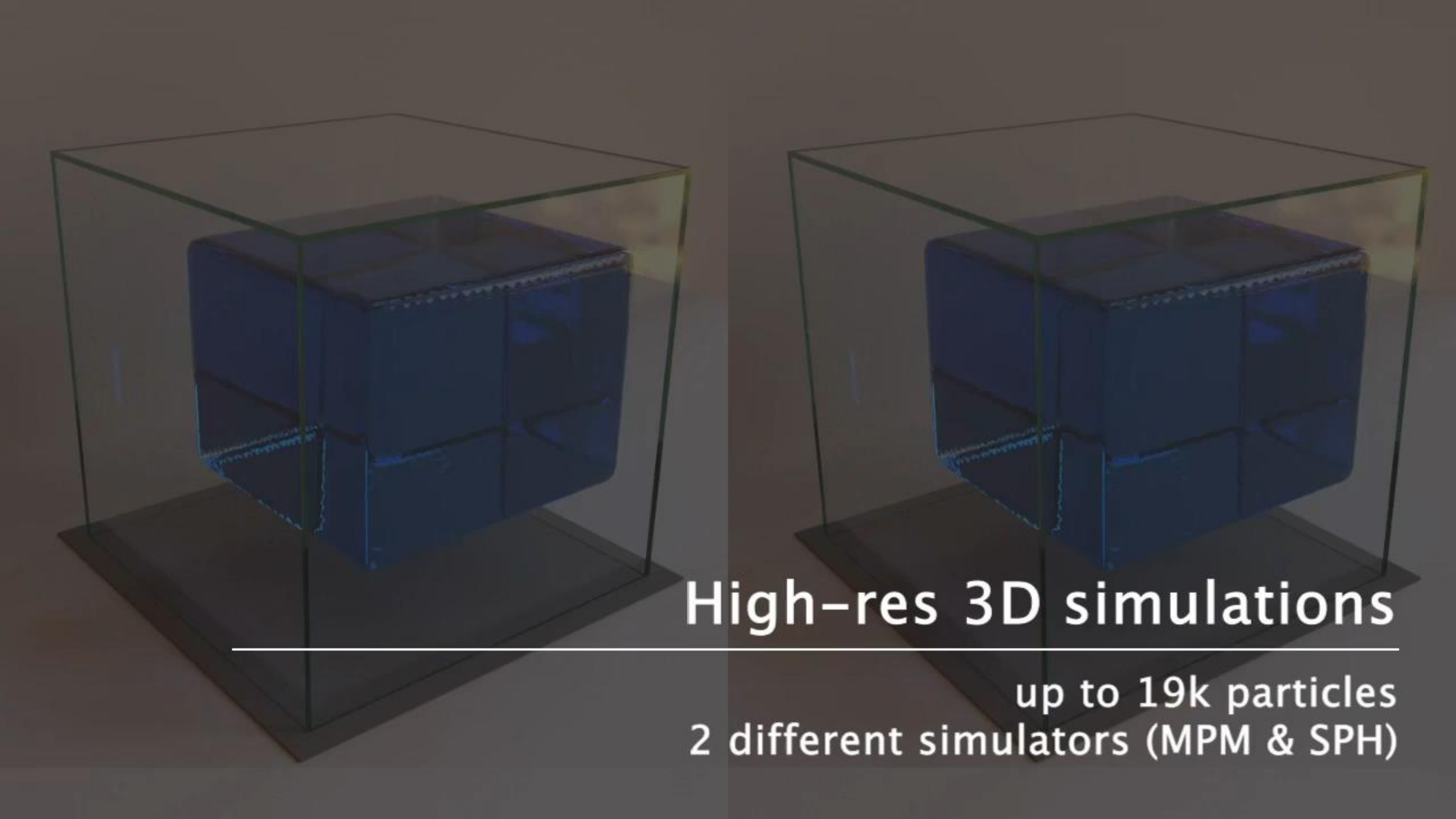
Graph
Algorithms

Sensor Nets?
Biology?!?

Graph Neural Networks

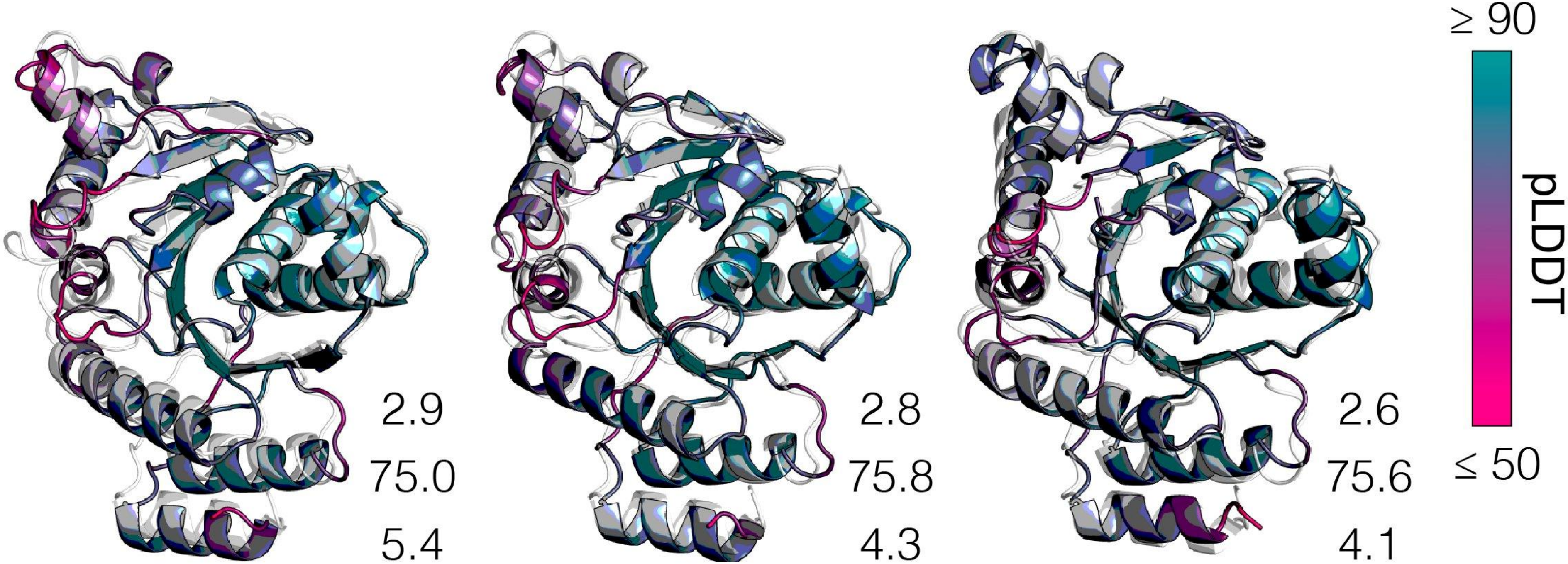
Roger Wattenhofer


vs.
Networks
Images

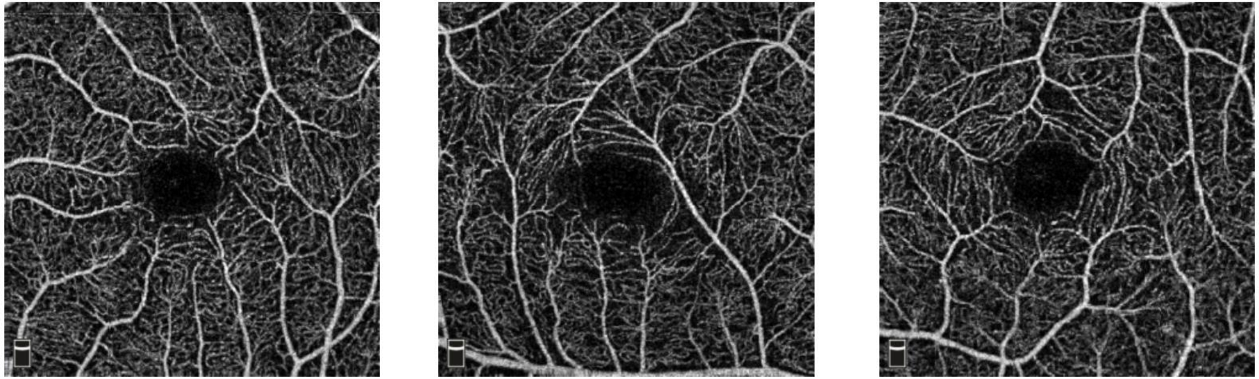

High-res 3D simulations
up to 19k particles
2 different simulators (MPM & SPH)

Language models of protein sequences at the scale of evolution enable accurate structure prediction

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, Alexander Rives



EULER CHARACTERISTIC SURFACES

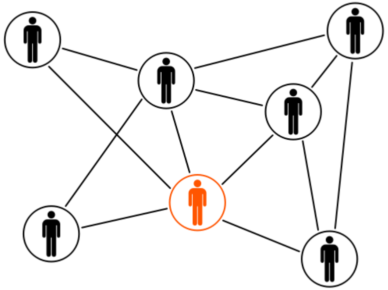


(a) Control: No disease

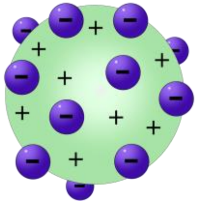
(b) NoDR: Diabetes, but no retinopathy.

(c) DR: Diabetes with retinopathy.

social networks

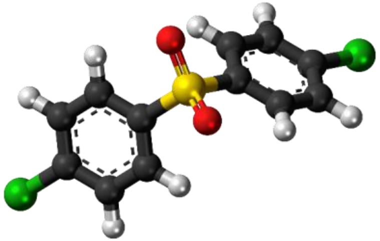


chemo-informatics



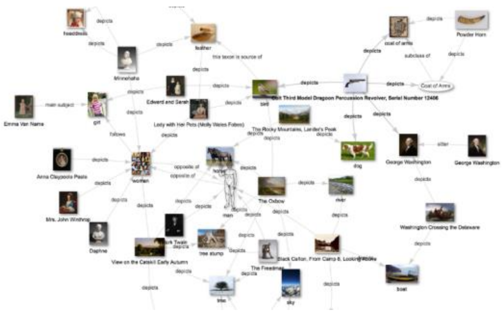
question answering systems

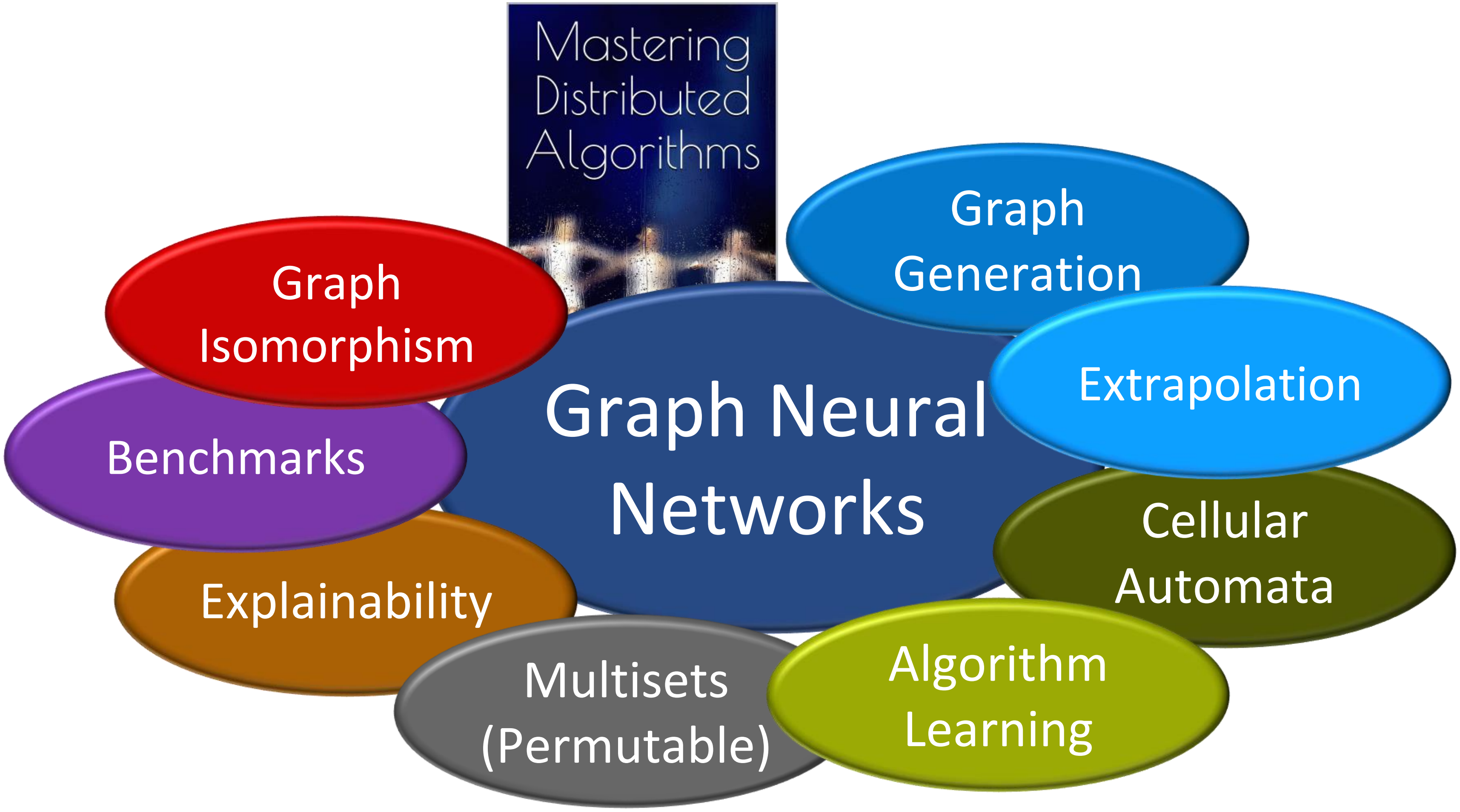
molecule recognition



recommender systems

knowledge graphs




Mastering Distributed Algorithms
Graph Isomorphism
Graph Generation
Graph Neural Networks
Extrapolation
Benchmarks
Cellular Automata
Explainability
Multisets (Permutable)
Algorithm Learning

An Introduction to Graph Neural Networks from a Distributed Computing Perspective

Pál András Papp and Roger Wattenhofer

ETH Zürich, Switzerland
{apapp,wattenhofer}@ethz.ch

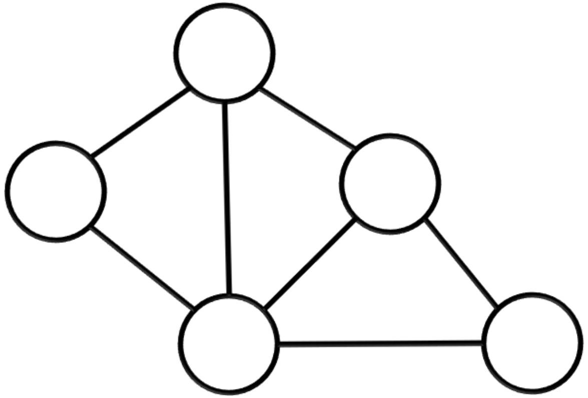
Abstract. The paper provides an introduction into the theoretical expressiveness of graph neural networks. We discuss the basic properties and main applications of standard GNN models, and we show how these constructions are both upper and lower bounded in expressive power by the Weisfeiler-Lehman test. We then outline a wide variety of approaches to increase the expressiveness of GNNs above this theoretical limit, and discuss the strengths and weaknesses of these methods.

GNNs vs. Distributed Computing

Distributed Computing (Message Passing)

Nodes communicate with neighbors by sending messages.

In each synchronous round, every node sends a message to its neighbors.

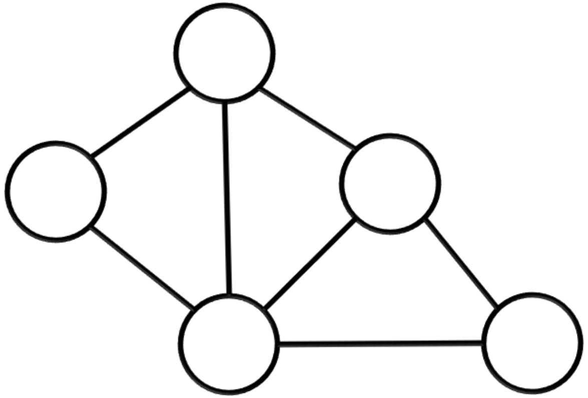


```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

Graph Neural Networks

Nodes communicate with neighbors by sending messages.

In each synchronous round, every node sends a message to its neighbors.



```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

DC Track

"Designed" algorithm

Usually node IDs

Individual messages

Solve graph problems like coloring or routing

```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

ML Track

"Learned" parameters

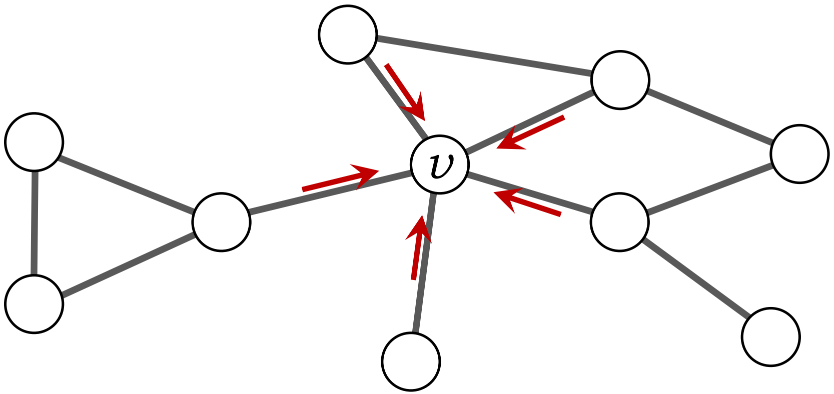
Usually node features

Aggregated messages

Solve classification (node, edge, graph)

How Do GNNs Work?

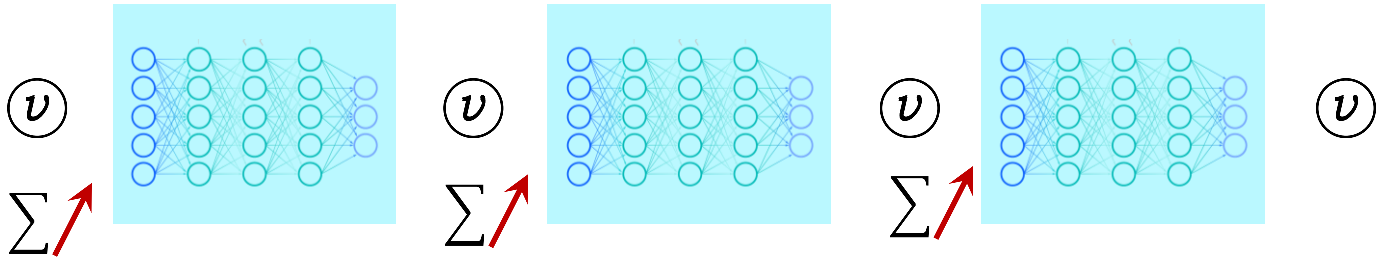
Graph Neural Networks



$$a_v = \textsc{Aggregate}\left(\{\{\, h_u \mid u \in N(v) \,\}\}\right)$$

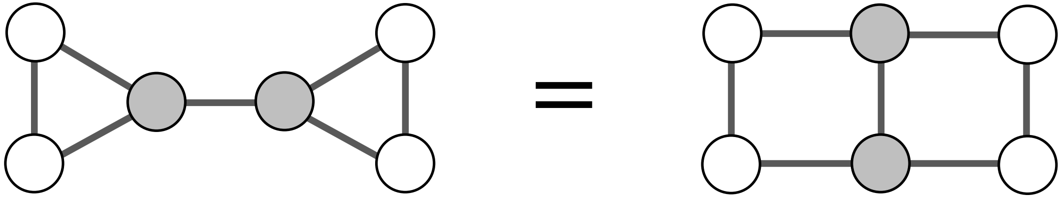
$$h_v^{(t+1)} = \textsc{Update}\left(h_v,\, a_v\right)$$

Graph Neural Networks

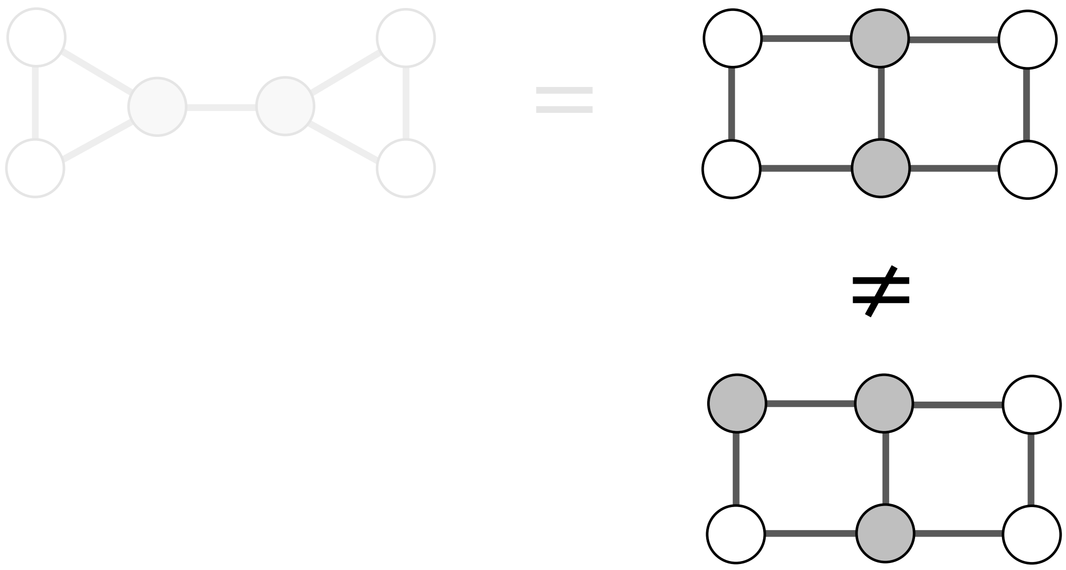


Any Limitations?

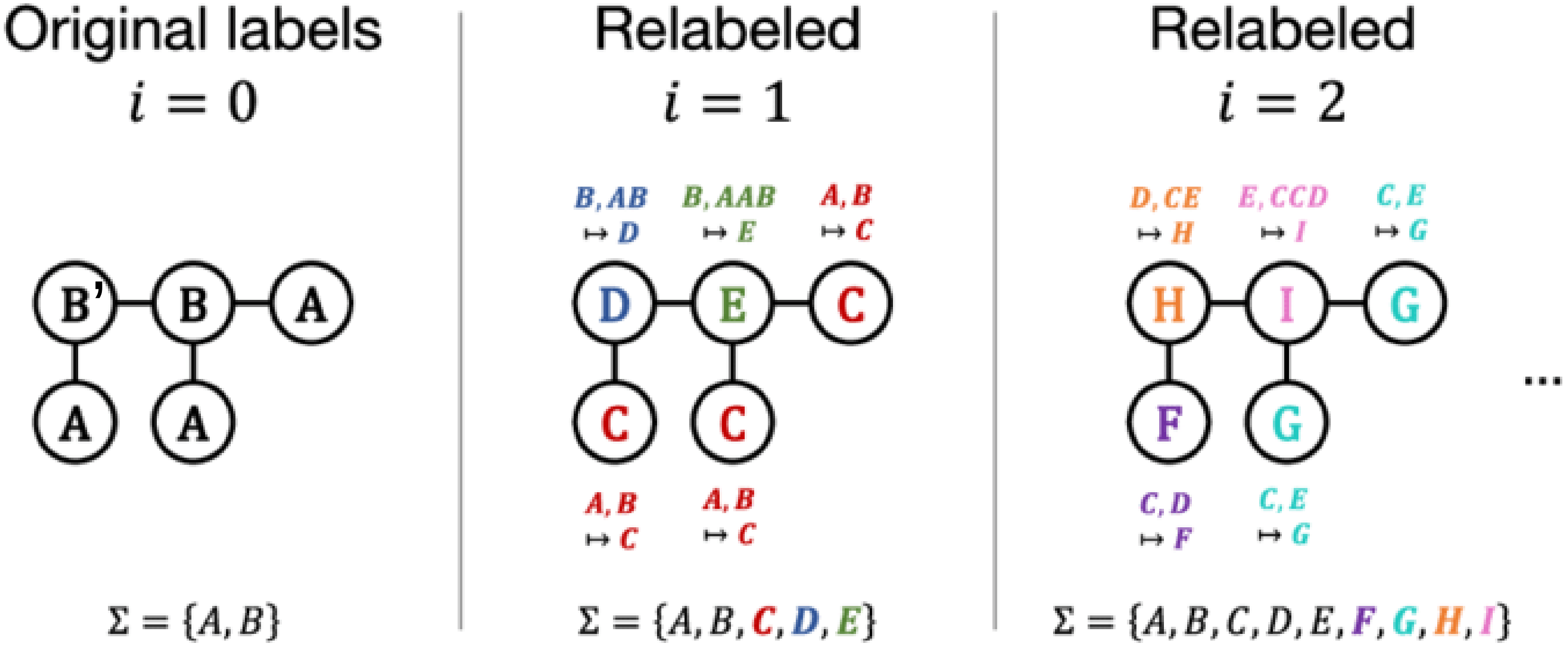
Graph Neural Networks



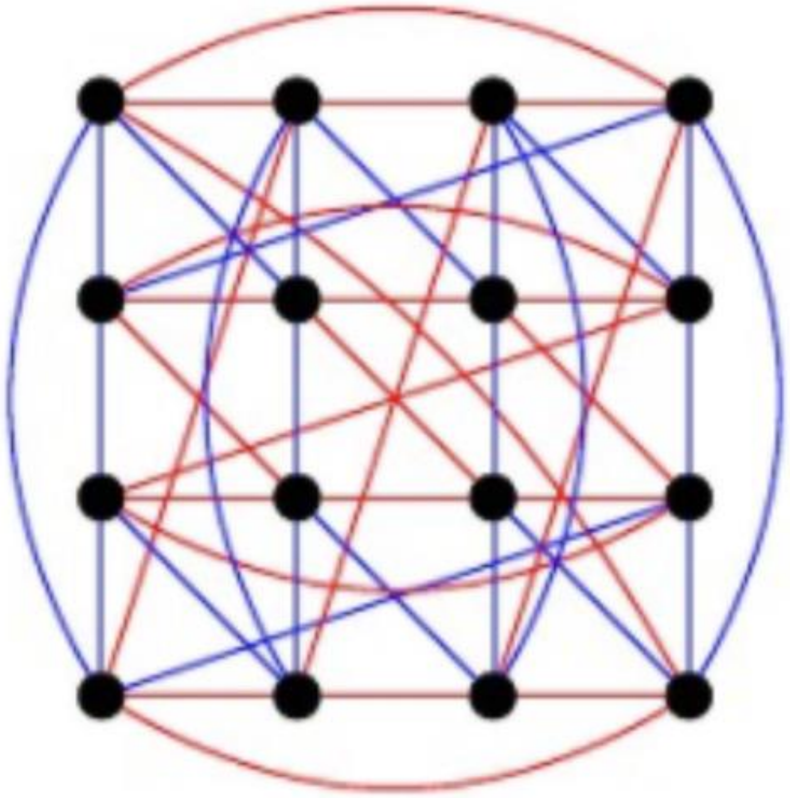
Graph Neural Networks

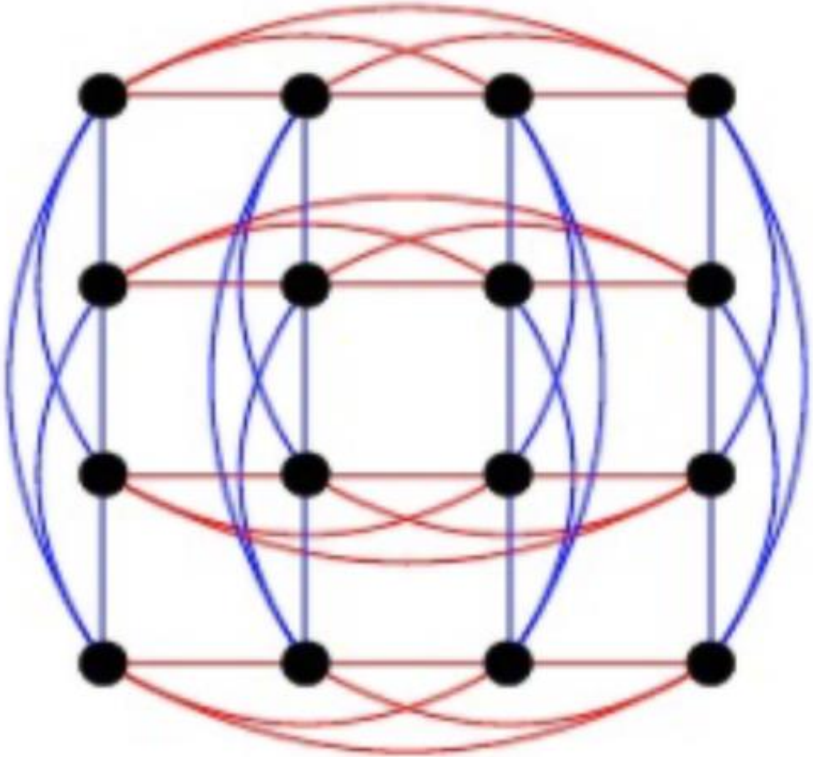


Weisfeiler-Lehman Graph Isomorphism Test

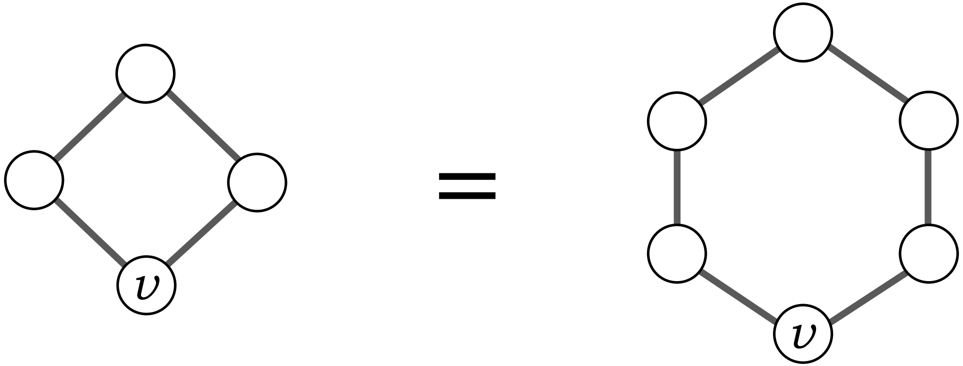


Shrikande vs. Rooks





GNNs Fail on e.g. Cycles



DC Track

aggregation

local

congest

each round:
every node:
1. send msgs
2. rcv msgs
3. compute

ML Track

oversmoothing

underreaching

oversquashing

More Expressive GNNs?

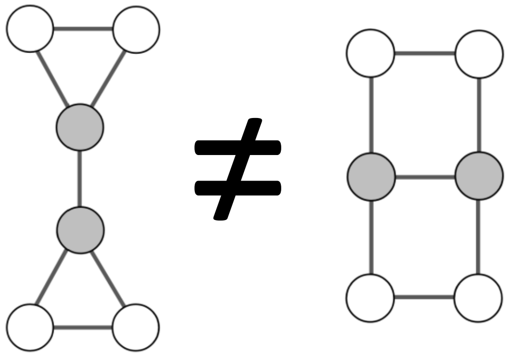
DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks

Pál András Papp
ETH Zurich
apapp@ethz.ch

Karolis Martinkus
ETH Zurich
martinkus@ethz.ch

Lukas Faber
ETH Zurich
lfaber@ethz.ch

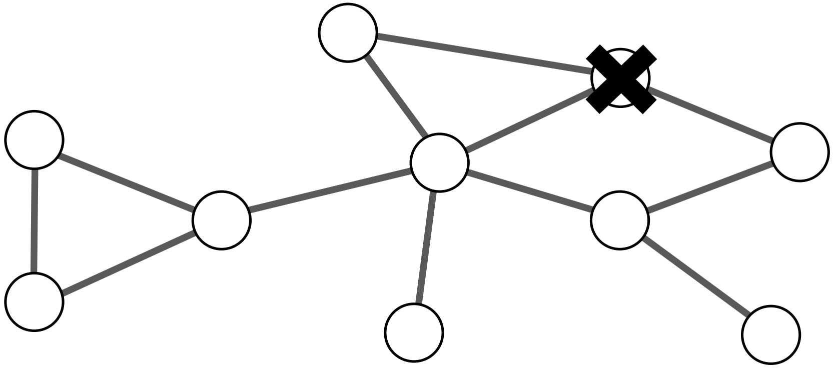
Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch



GNNs with Dropouts

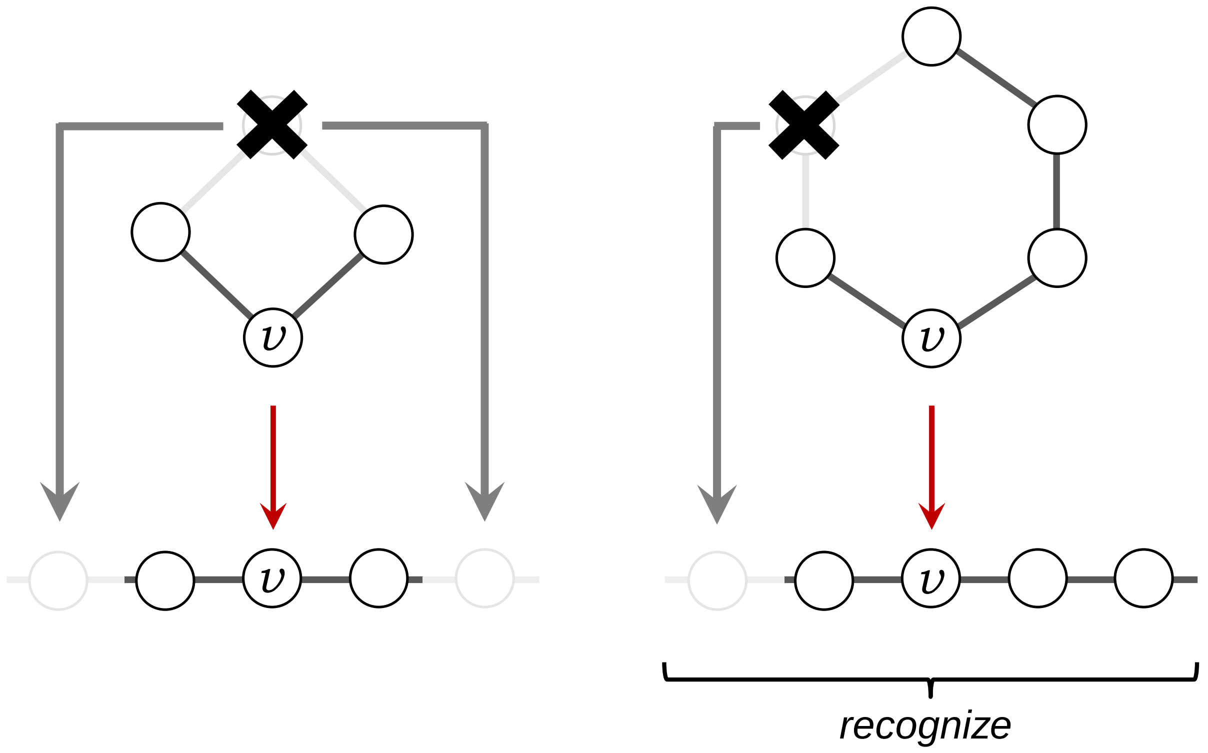
Multiple runs of the GNN

Each node removed with probability *p* independently

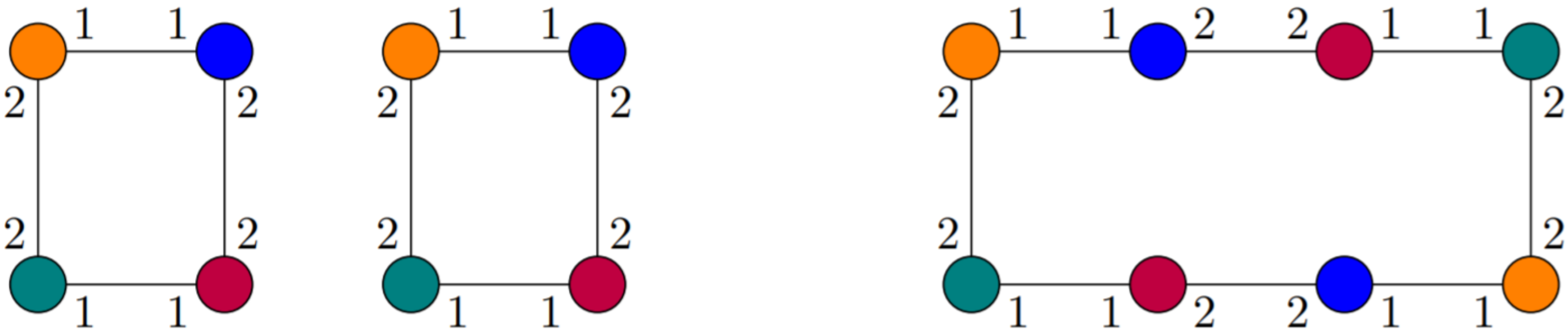


Run #1

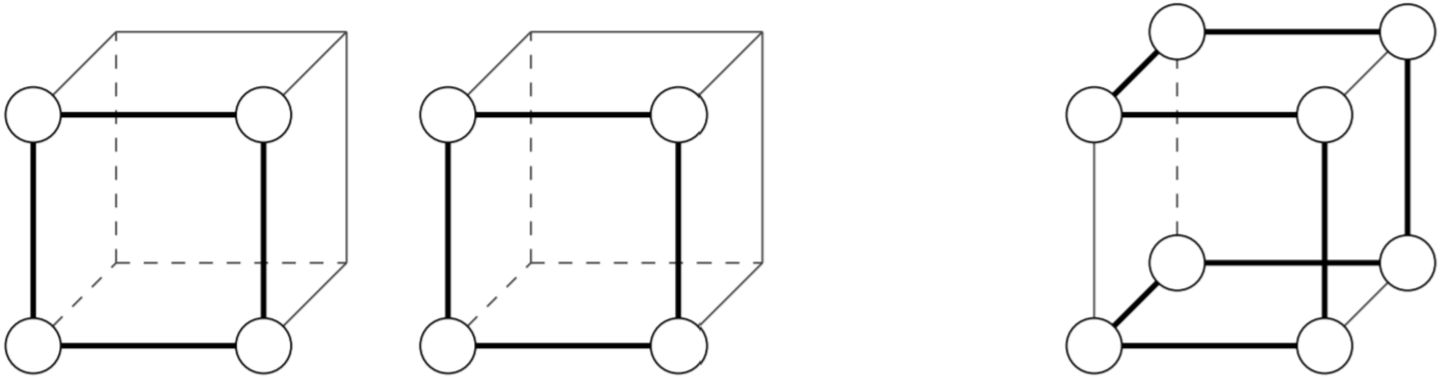
GNNs with Dropouts



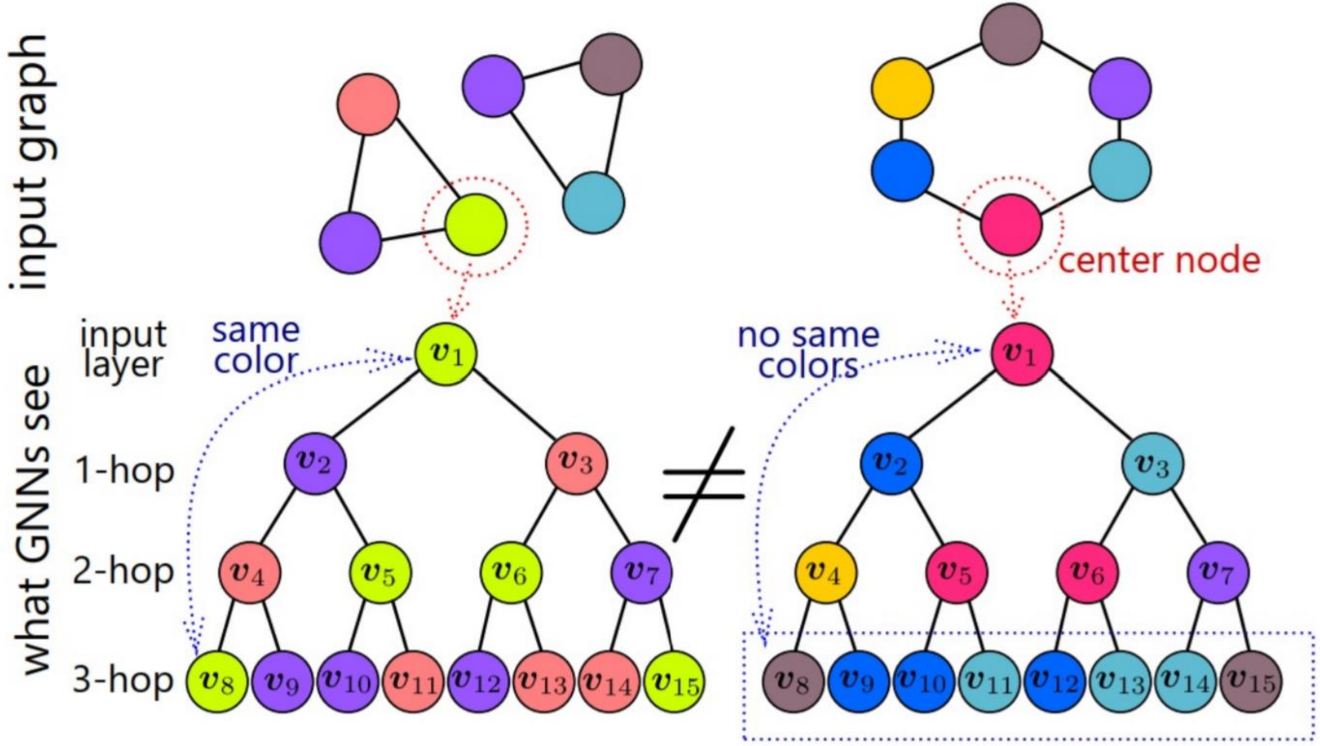
Port Numbers



Angle Features

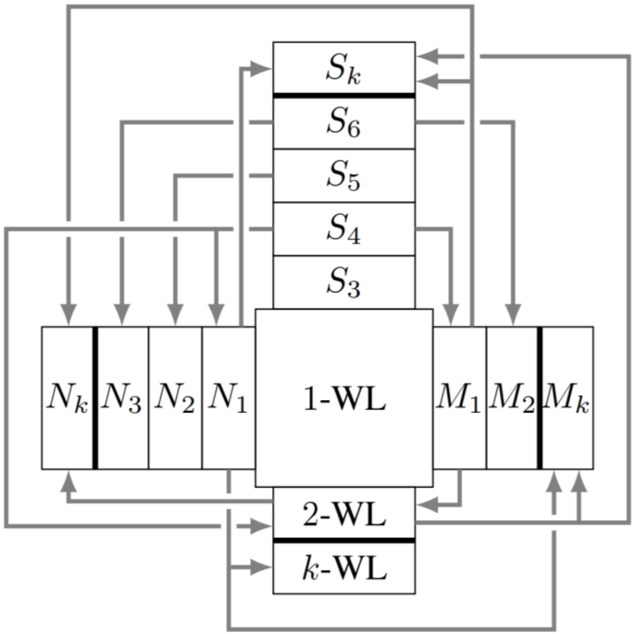


Random Features



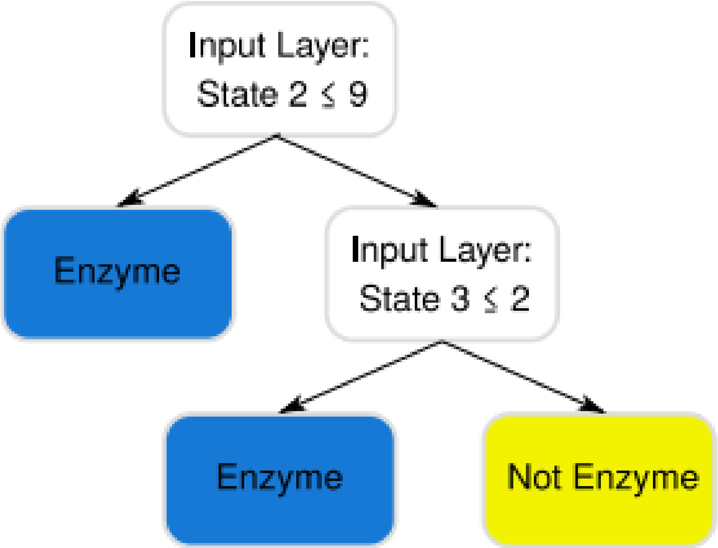
A Theoretical Comparison of Graph Neural Network Extensions

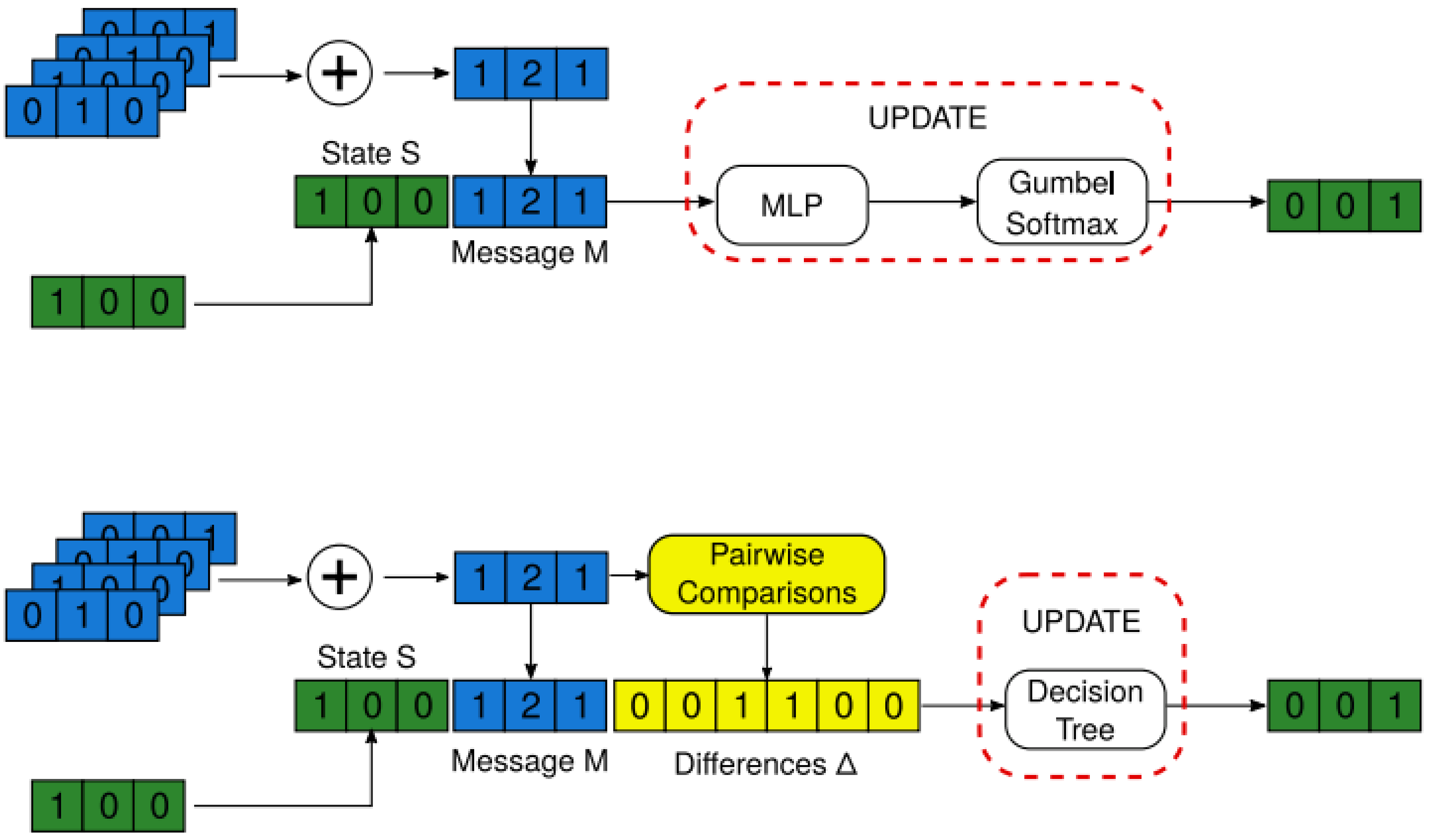
Pál András Papp [1] **Roger Wattenhofer** [1]



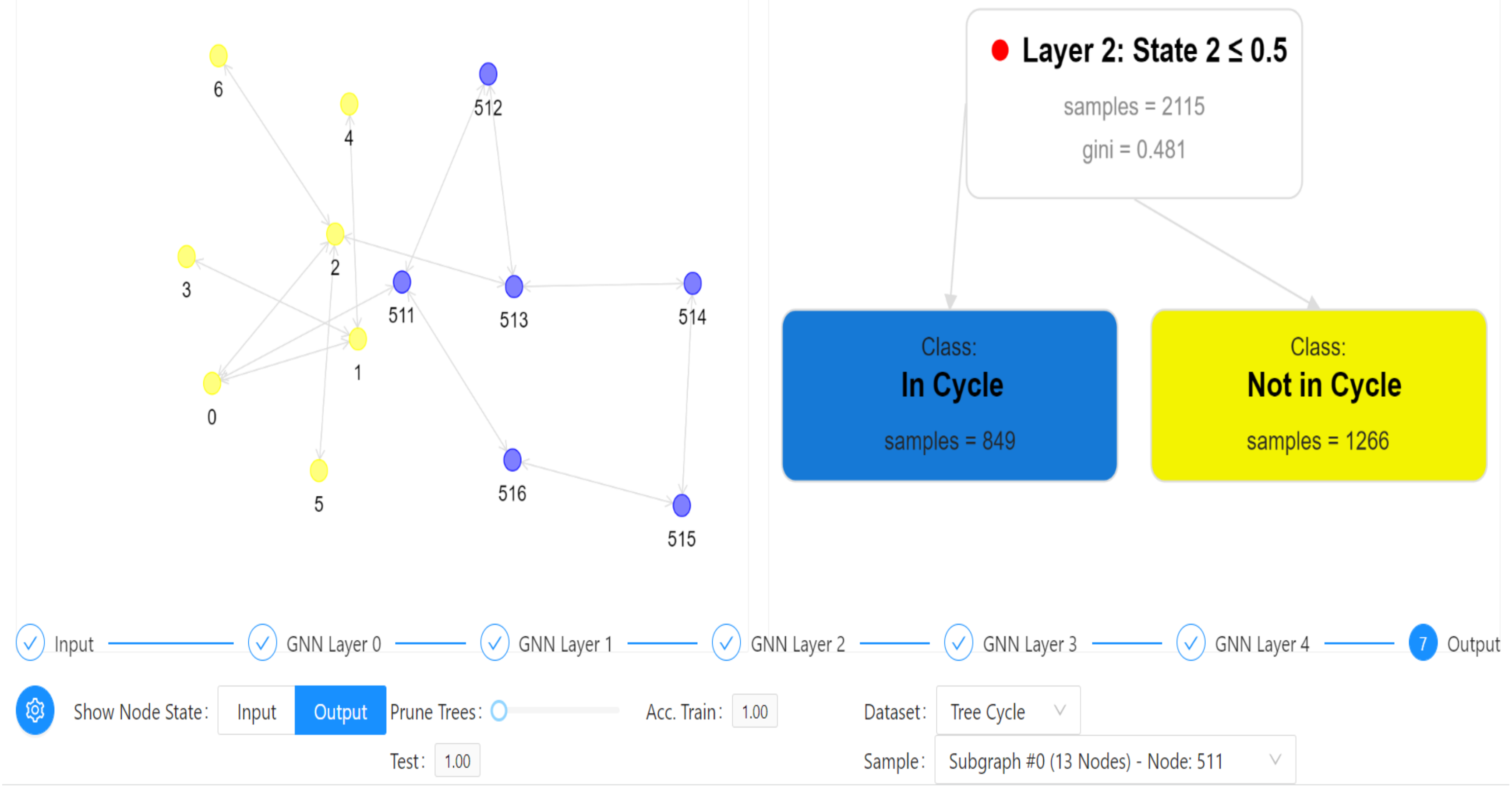
Explainable GNN

DT+GNN: A Fully Explainable Graph Neural Network using Decision Trees





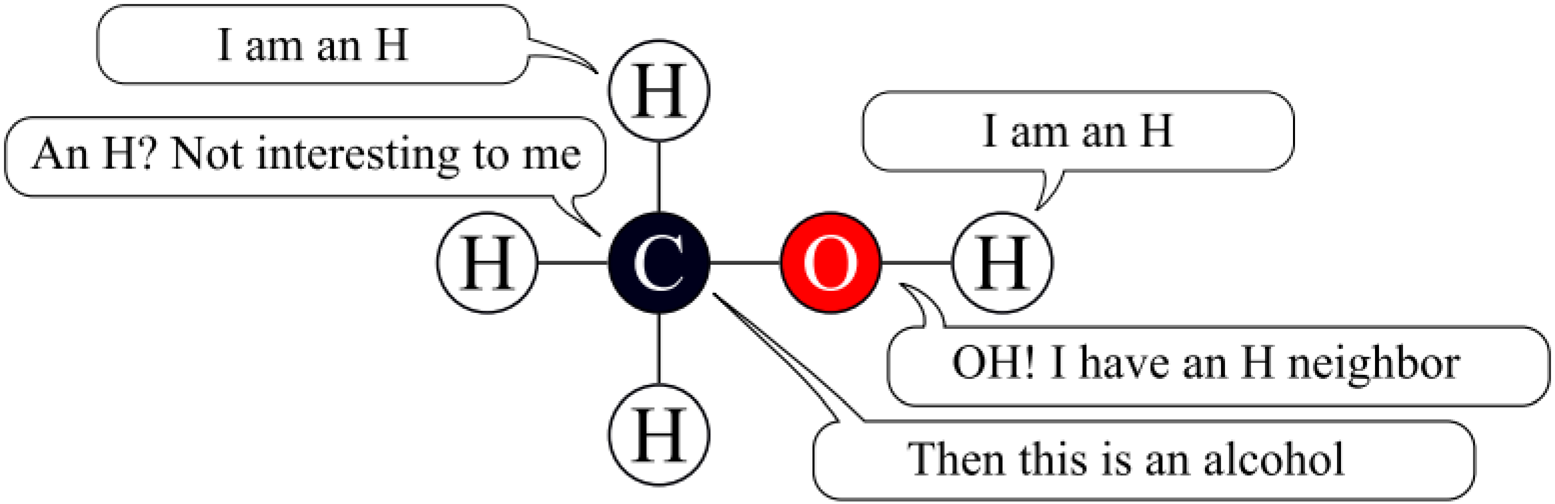
0 1 0
1 2 1
State S
1 0 0
1 2 1
Message M
1 0 0
UPDATE
MLP
Gumbel
Softmax
0 0 1
0 1 0
1 2 1
Pairwise
Comparisons
State S
1 0 0
1 2 1
0 0 1 1 0 0
Message M
Differences Δ
1 0 0
UPDATE
Decision
Tree
0 0 1



6
4
512
2
3
511
513
514
1
0
516
5
515
Layer 2: State 2 ≤ 0.5
samples = 2115
gini = 0.481
Class:
In Cycle
samples = 849
Class:
Not in Cycle
samples = 1266
Input
GNN Layer 0
GNN Layer 1
GNN Layer 2
GNN Layer 3
GNN Layer 4
7
Output
Show Node State:
Input
Output
Prune Trees:
Acc. Train:
1.00
Test:
1.00
Dataset:
Tree Cycle
Sample:
Subgraph #0 (13 Nodes) - Node: 511

Asynchronous GNN

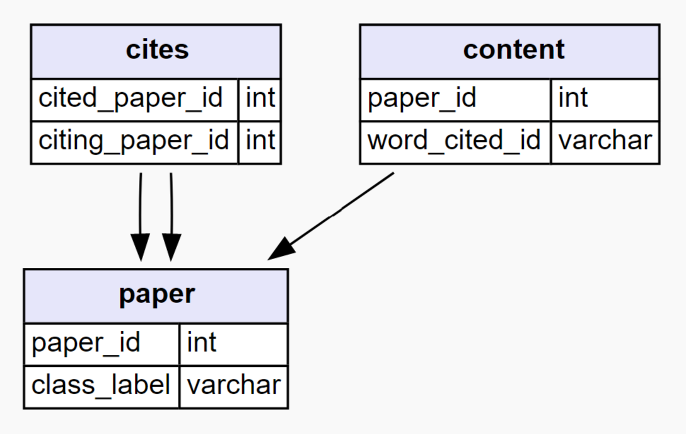
Asynchronous Neural Networks for Learning in Graphs

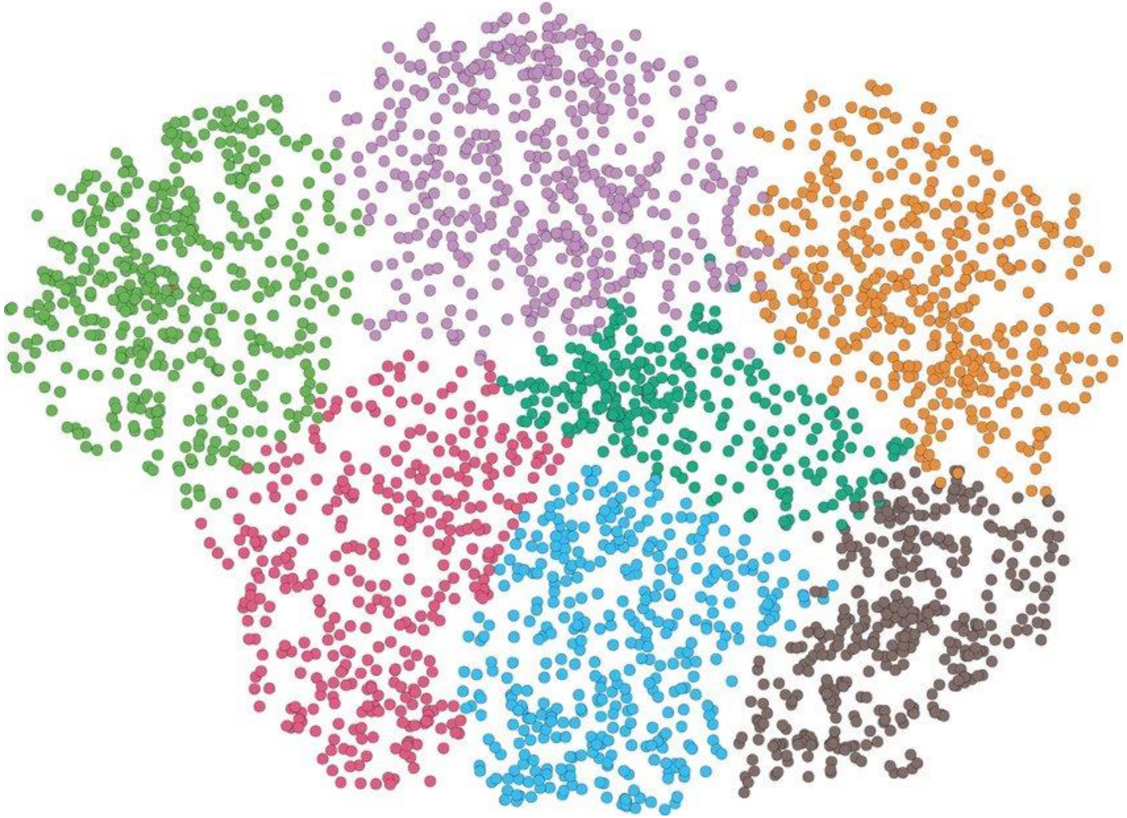


Benchmarks

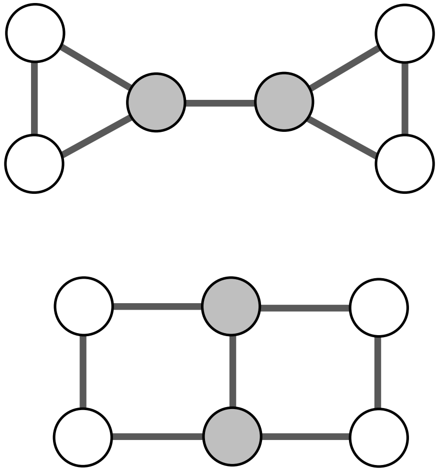

BAD DATA

Example: CORA Benchmark





Example: CORA Benchmark

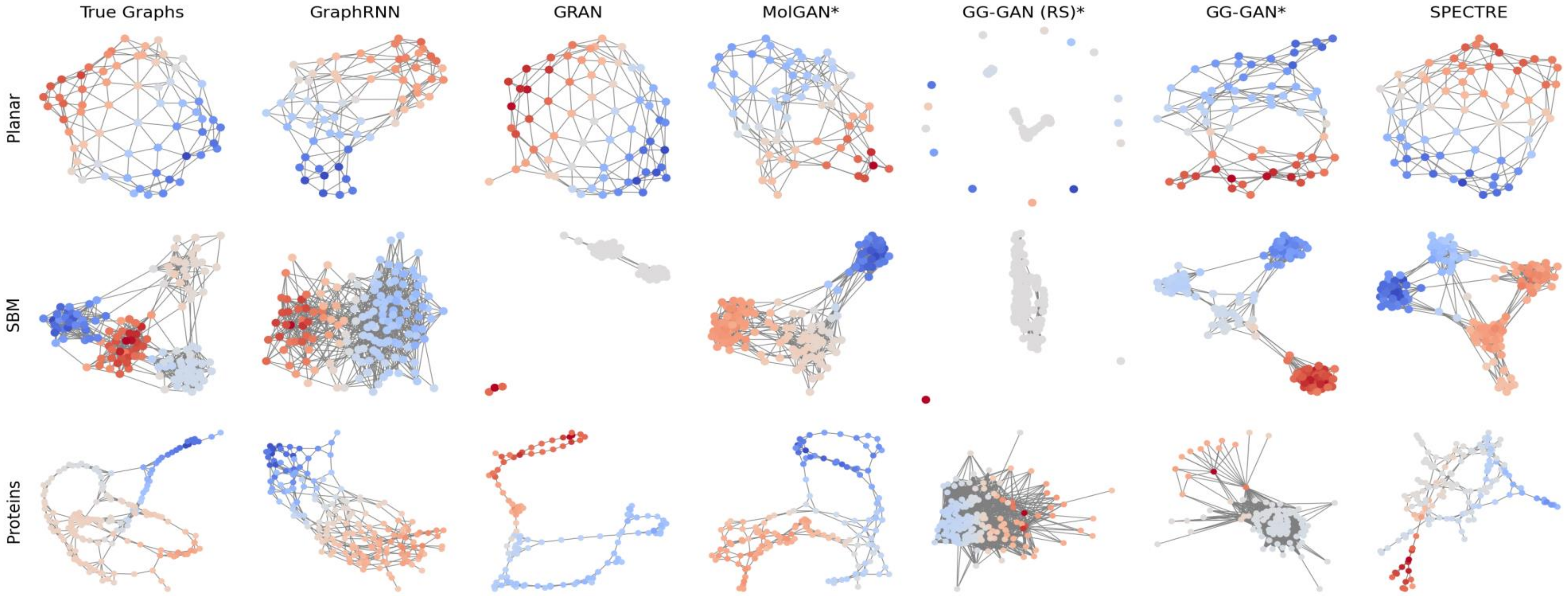


Title	Keywords	Neighbor Labels	Neighbor Keywords
Primes is in P	…	Crypto, …	…

Graph Generation

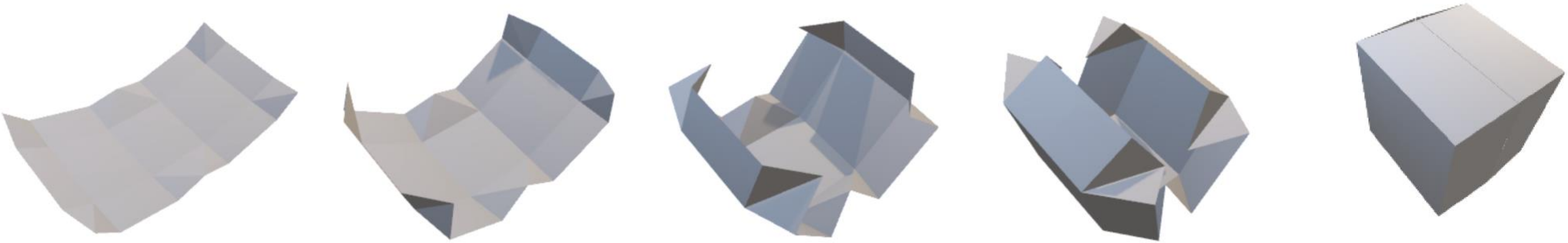
SPECTRE : Spectral Conditioning Helps to Overcome the Expressivity Limits of One-shot Graph Generators

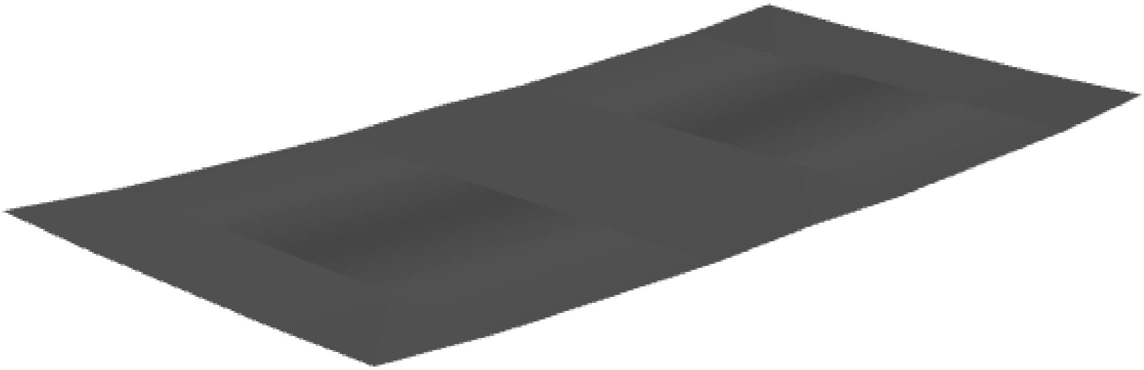
Karolis Martinkus [1] **Andreas Loukas** [*2] **Nathanaël Perraudin** [*3] **Roger Wattenhofer** [1]



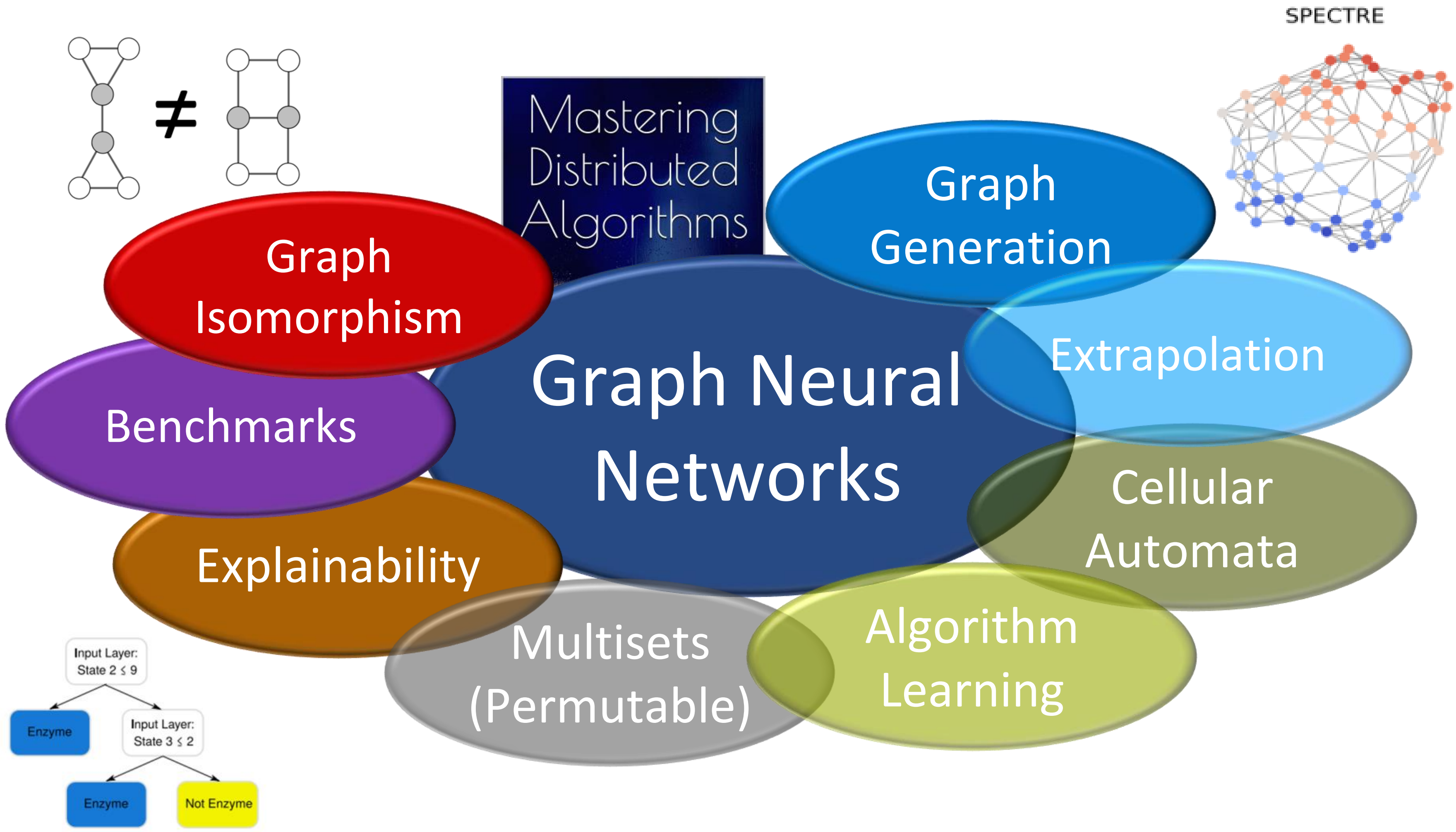
Automating Rigid Origami Design

Jeremia Geiger, Karolis Martinkus, Oliver Richter, Roger Wattenhofer





Summary



SPECTRE
Mastering Distributed Algorithms
Graph Isomorphism
Graph Generation
Graph Neural Networks
Extrapolation
Benchmarks
Cellular Automata
Explainability
Multisets (Permutable)
Algorithm Learning
Input Layer: State 2 ≤ 9
Enzyme
Input Layer: State 3 ≤ 2
Enzyme
Not Enzyme

Thank You!

Questions & Comments?