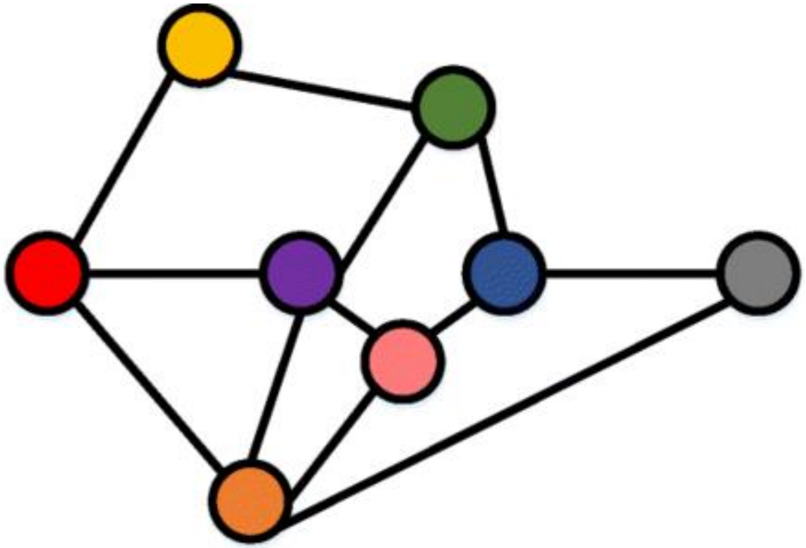
Networks, Dynamics, Algorithms … and Learning

Roger Wattenhofer

Graph Neural Networks



but first…

Learning Algorithms with Self-Play: A New Approach to the Distributed Directory Problem

Pankaj Khanchandani
Cloud Technology
Adobe Systems, India
kpankaj@adobe.com

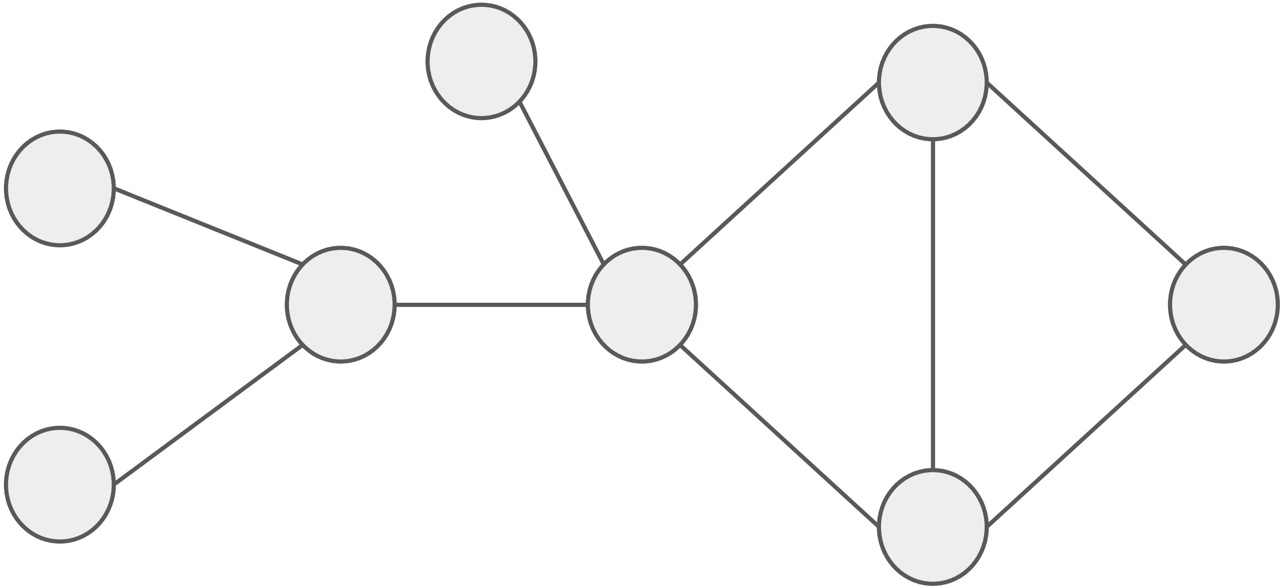
Oliver Richter, Lukas Rusch and Roger Wattenhofer
Department of Electrical Engineering and Information Technology
ETH Zurich, Switzerland
{richtero,ruschl,wattenhofer}@ethz.ch

***Abstract*—Many deep learning methods have been proposed recently to learn algorithms for combinatorial problems. However, most approaches focus on either supervised/imitation learning (the target algorithm is known) or single agent reinforcement learning (the input distribution is fixed). In some cases, however, the input distribution scales combinatorially as well and cannot easily be fully represented in a concise data set. In this paper, we propose a self-play approach to learn a *distributed directory protocol* to coordinate concurrent requests to a shared mobile resource among a network of nodes. The self-play is between two**

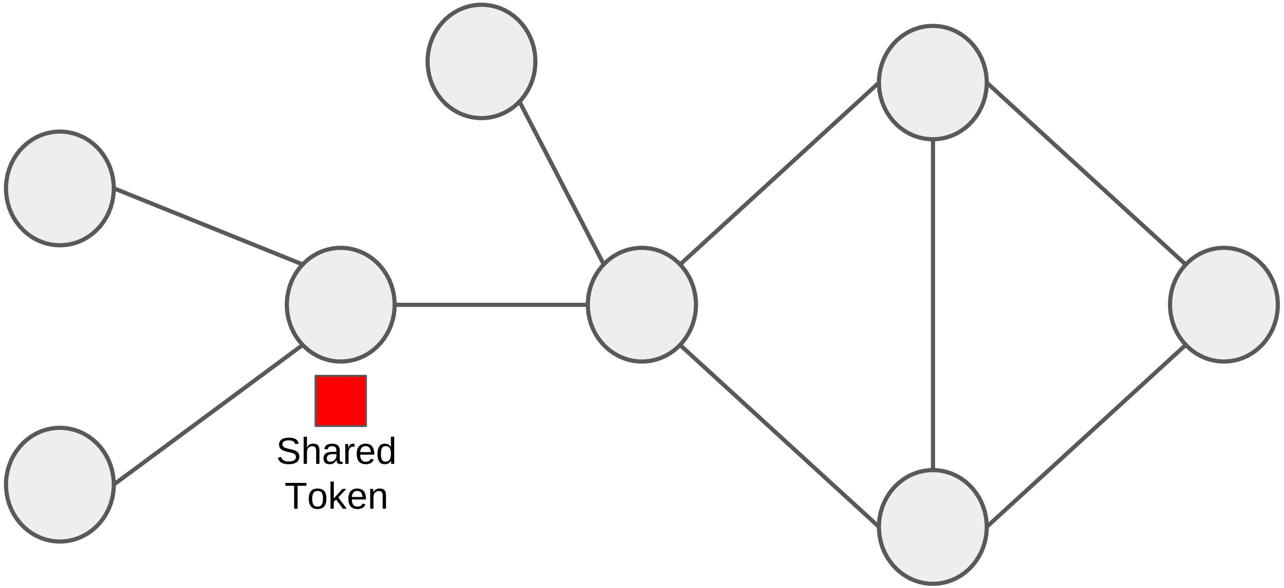
communication. While asymptotically optimal protocols exist for a few network topologies [1], many settings remain an open problem with no known best solution. We show that our approach performs on par with optimal protocols where such protocols exist and even empirically improves upon well known protocols by a large margin otherwise.

Further, we show that alternative learning approaches lead to sub-optimal protocols that can be exploited, while our self-play approach is robust against adversarial attacks.

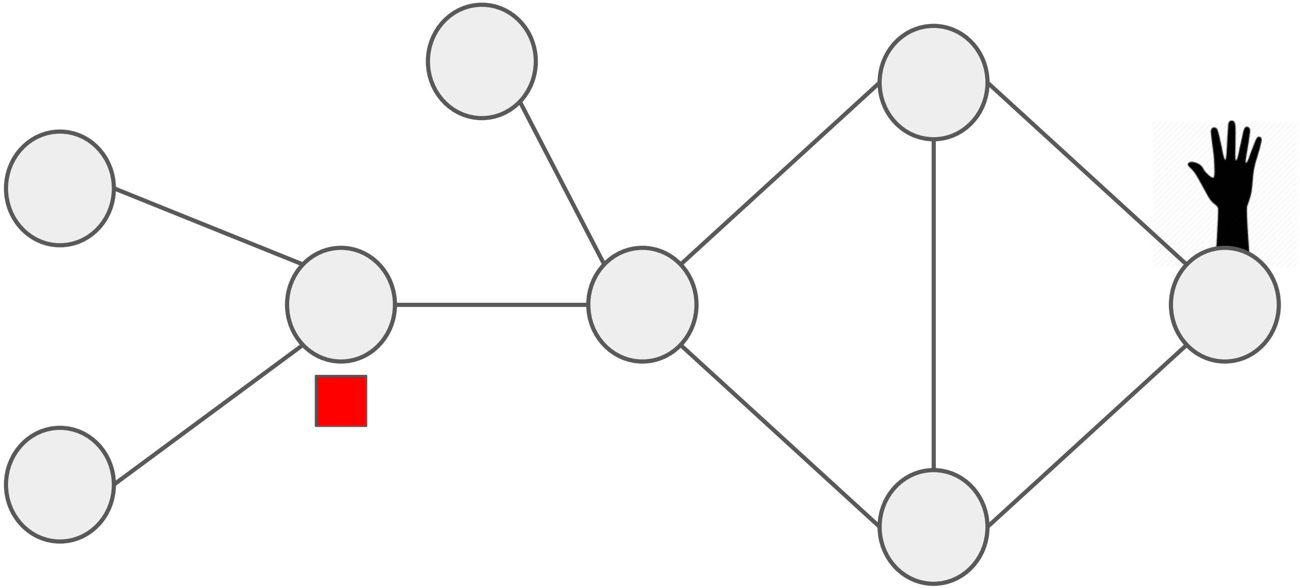
Distributed Directory



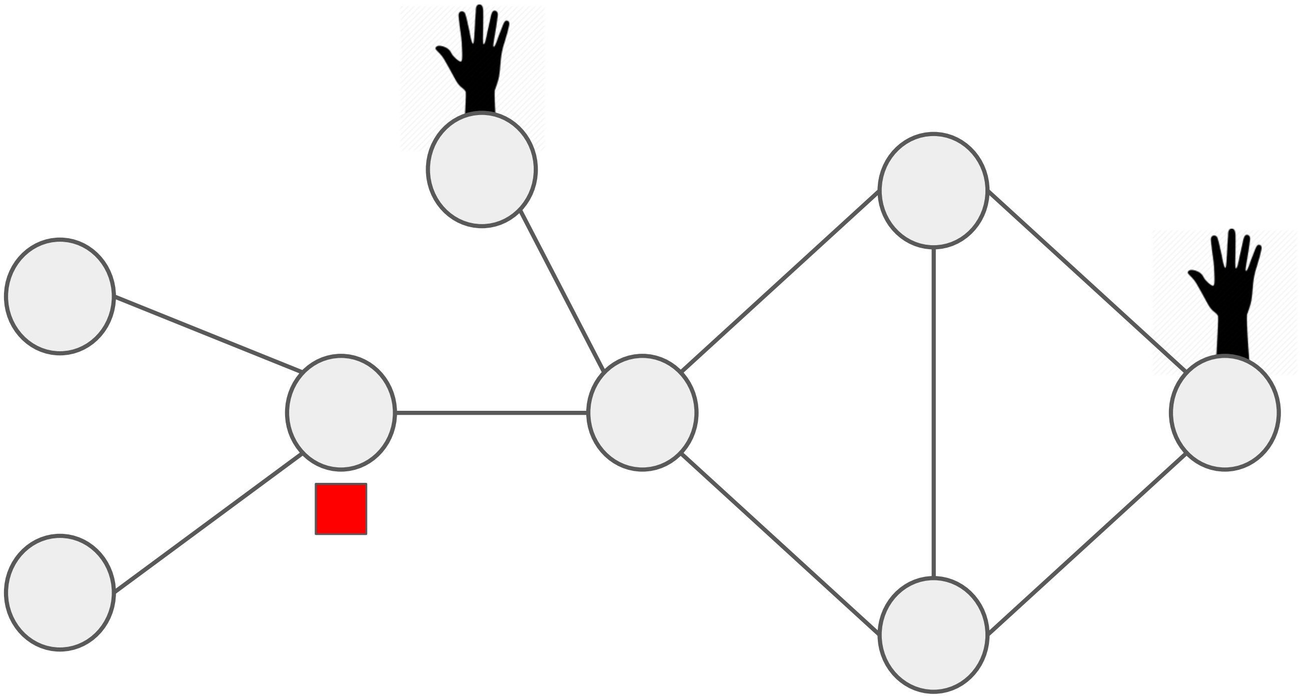
Distributed Directory



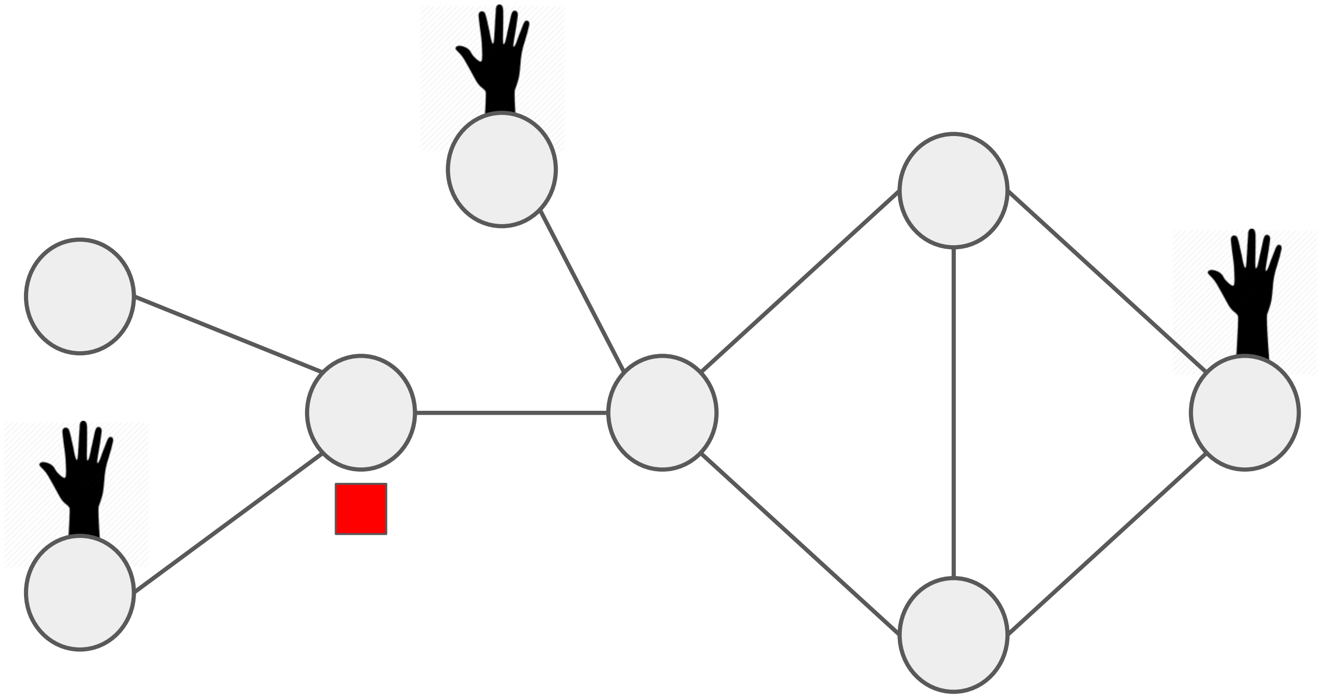
Distributed Directory



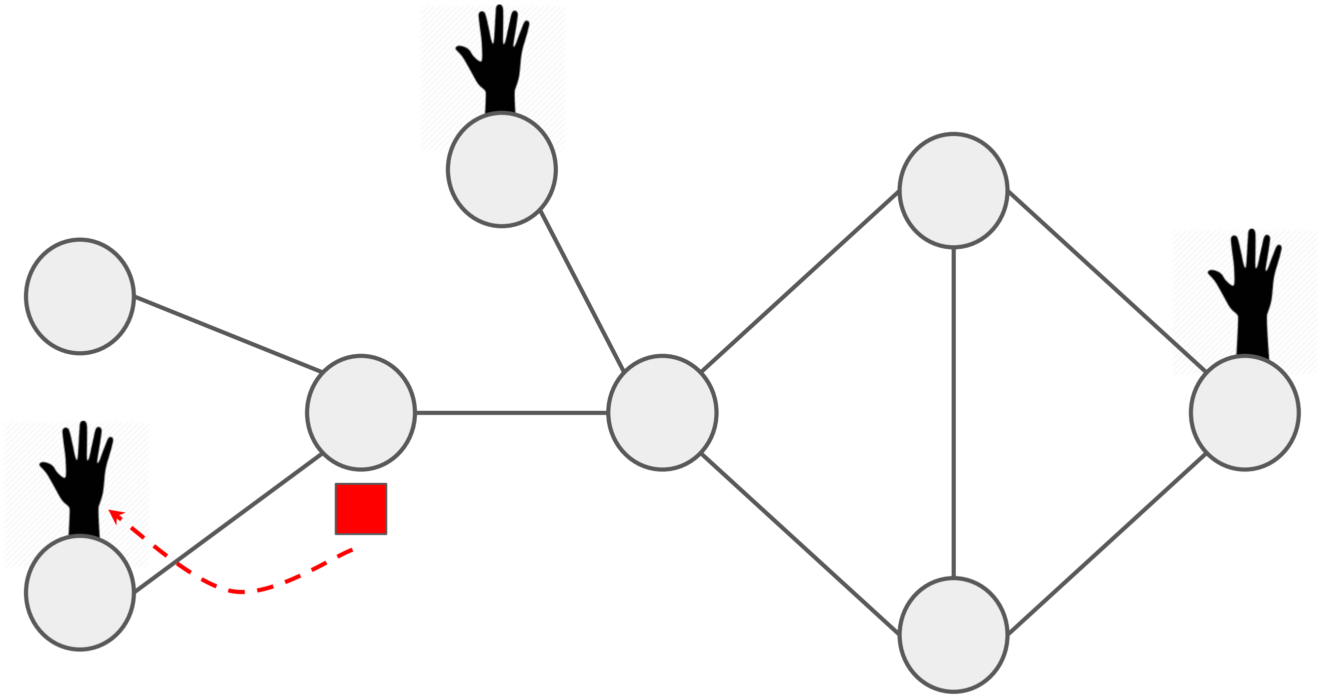
Distributed Directory



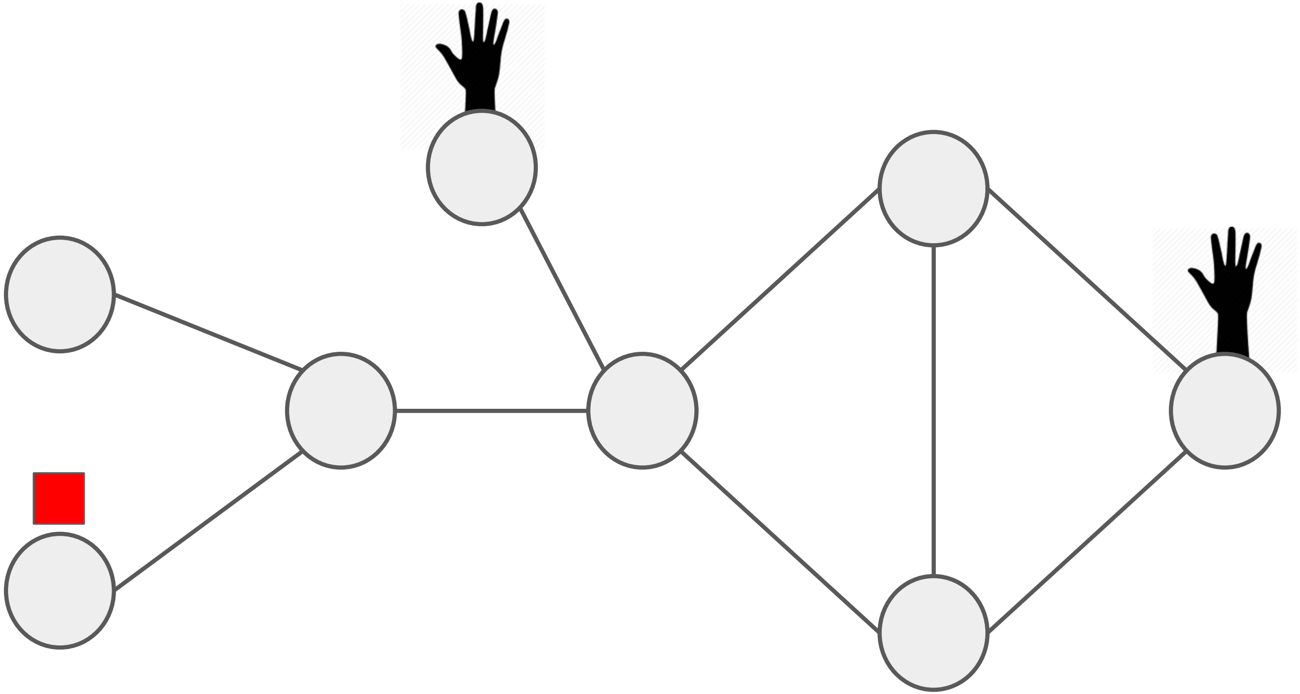
Distributed Directory



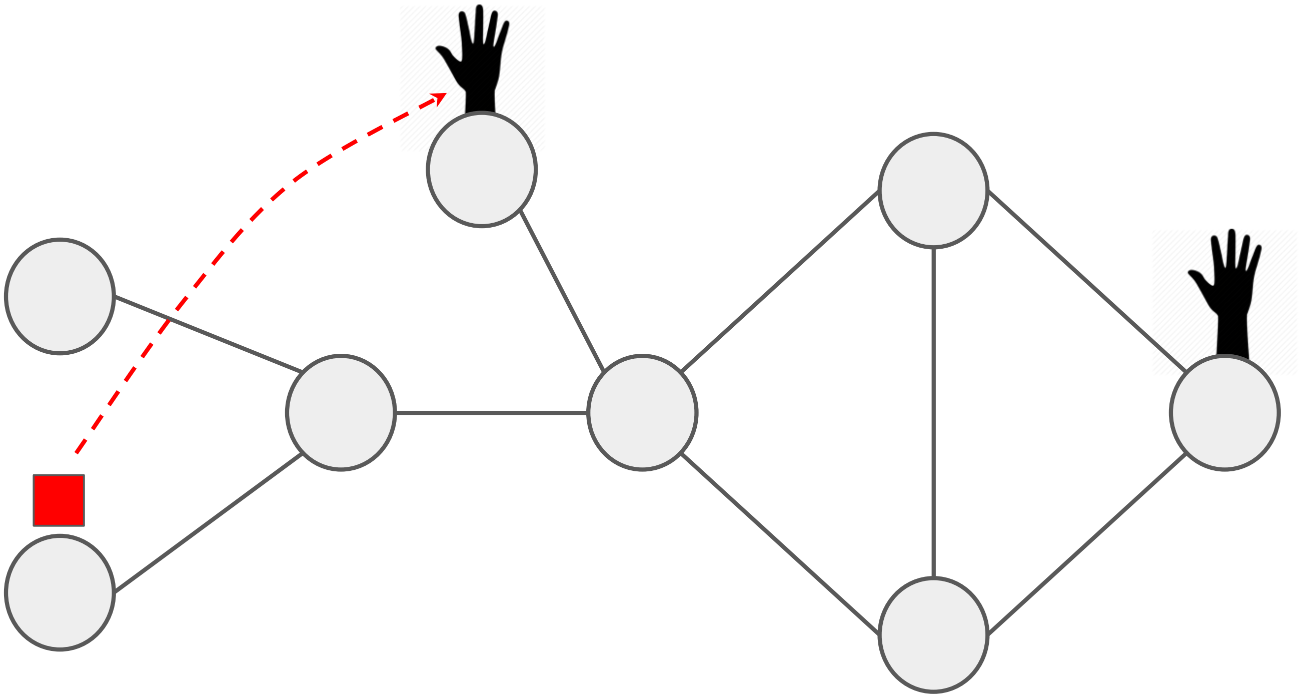
Distributed Directory



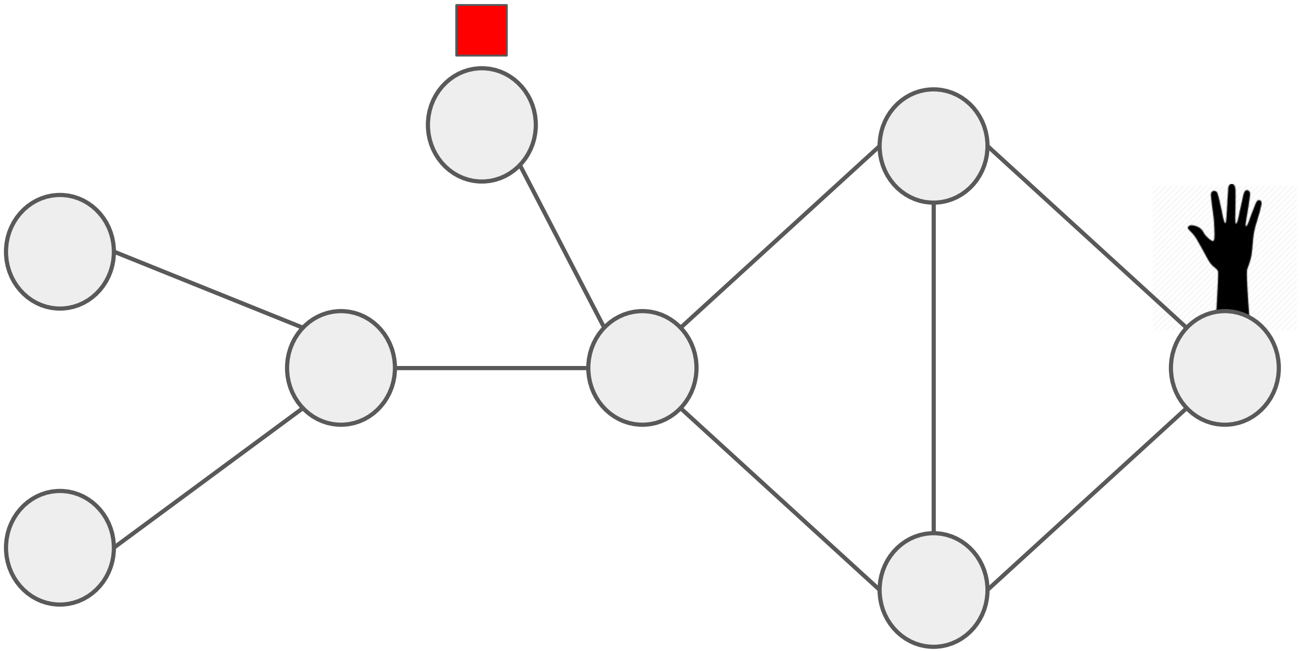
Distributed Directory



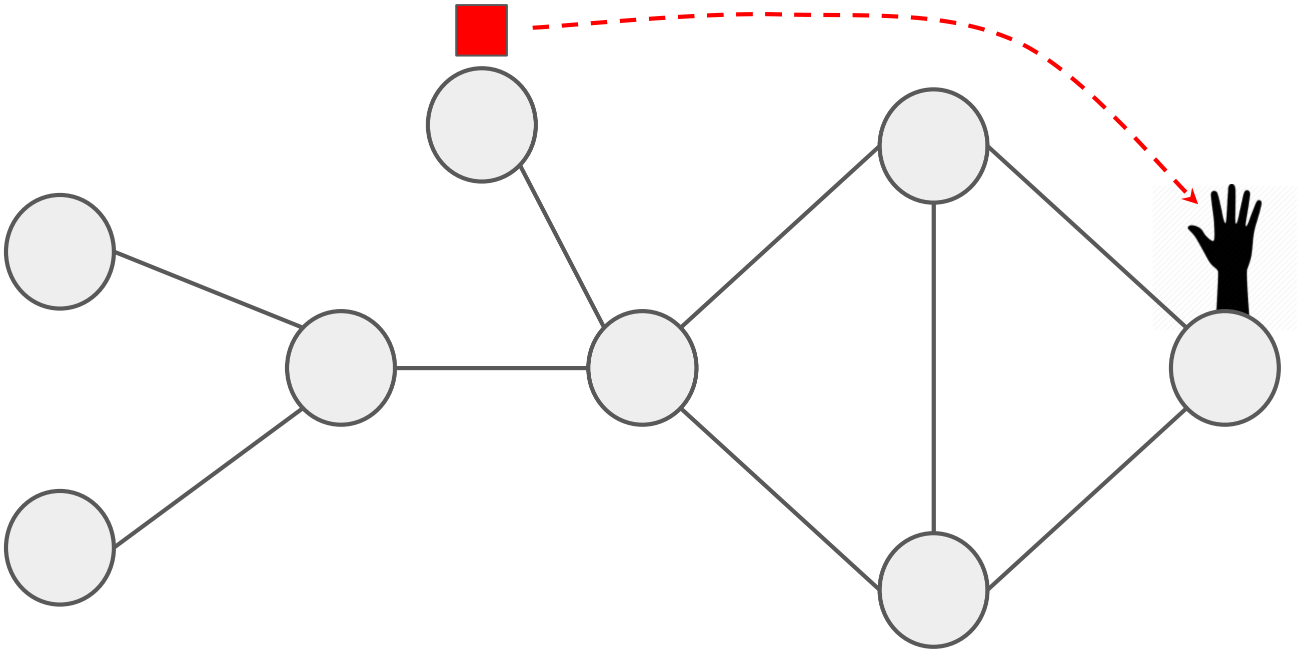
Distributed Directory



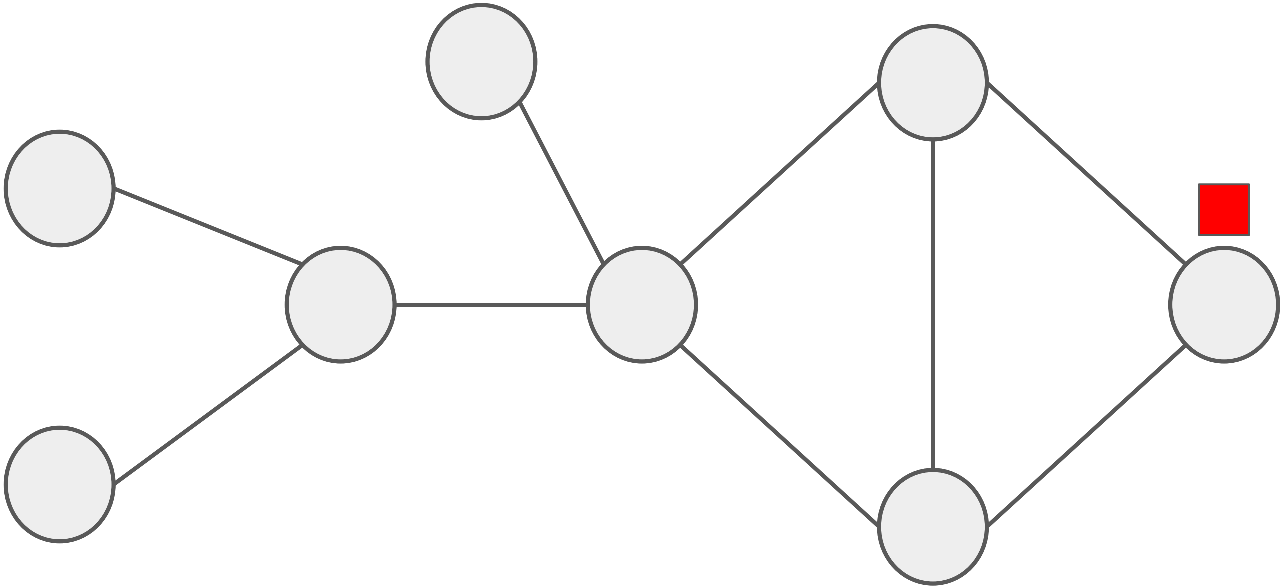
Distributed Directory



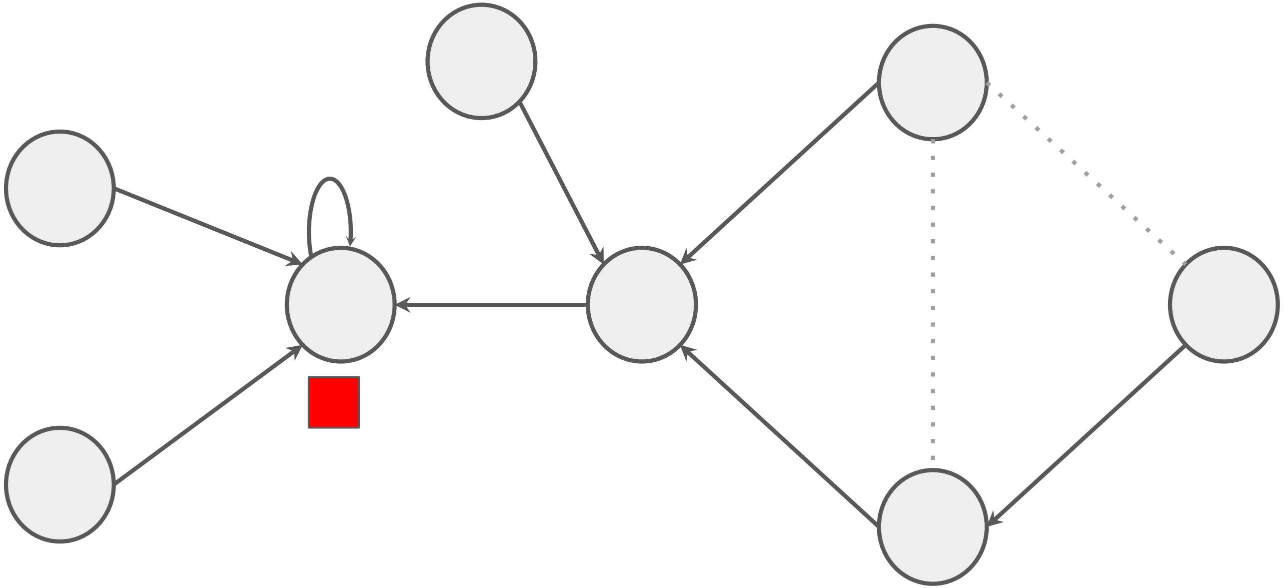
Distributed Directory



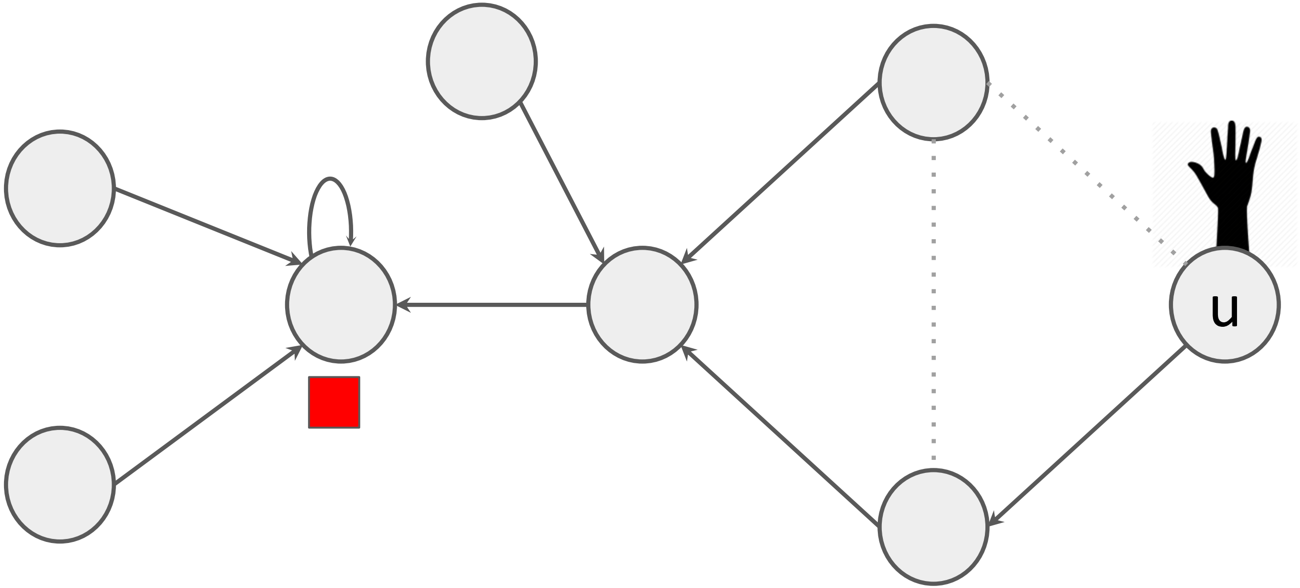
Distributed Directory



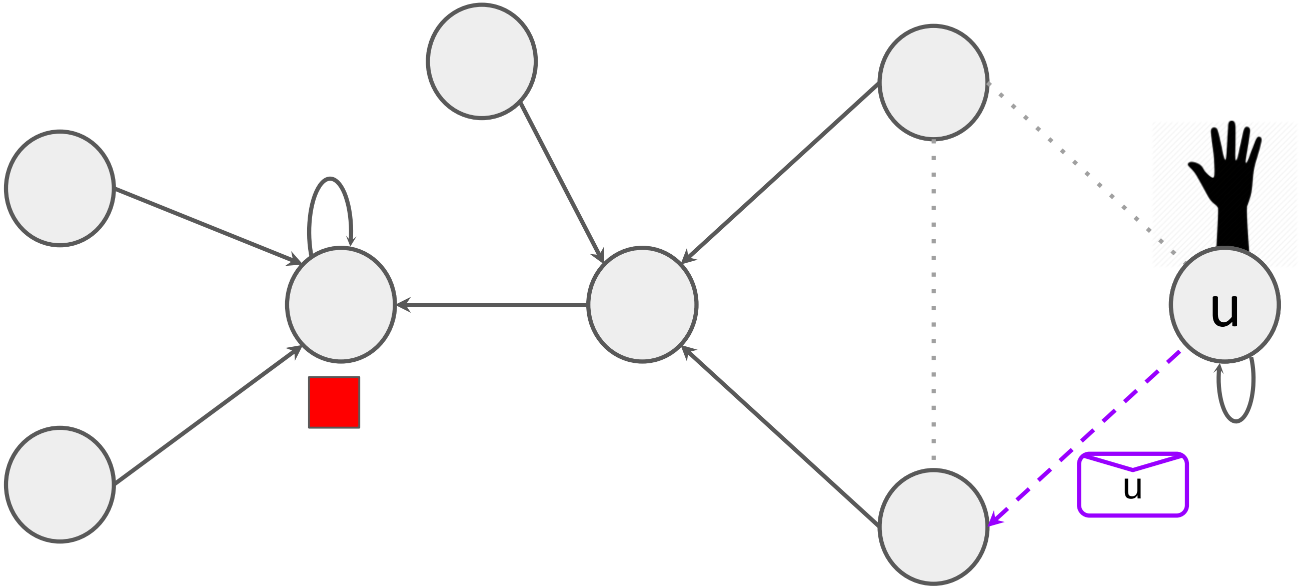
Arrow



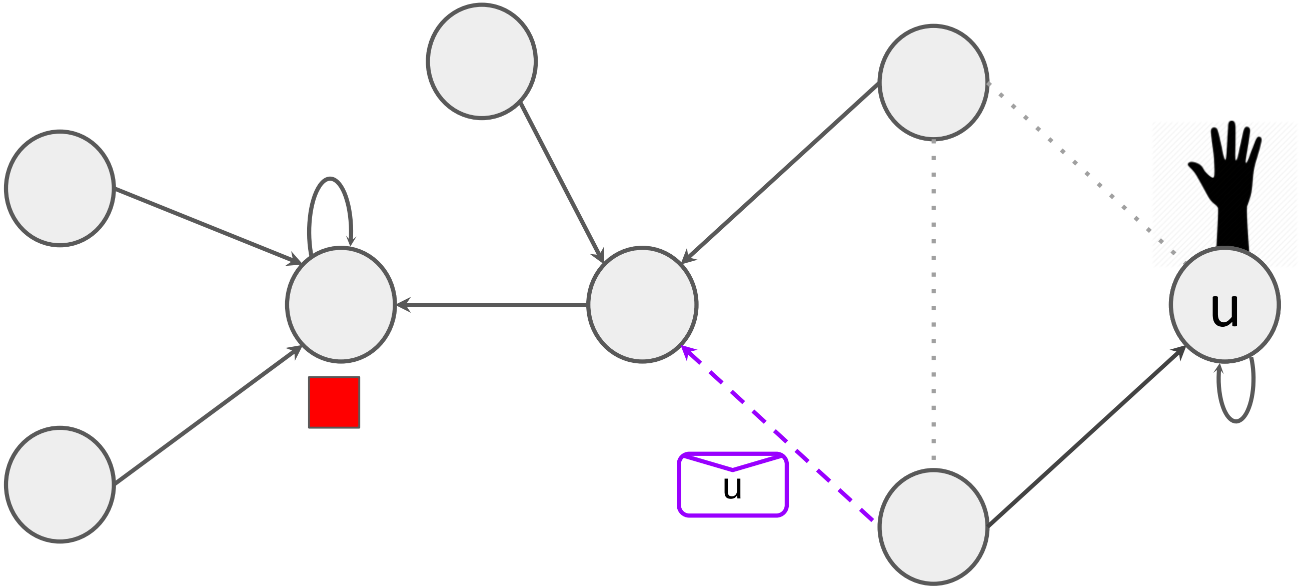
Arrow



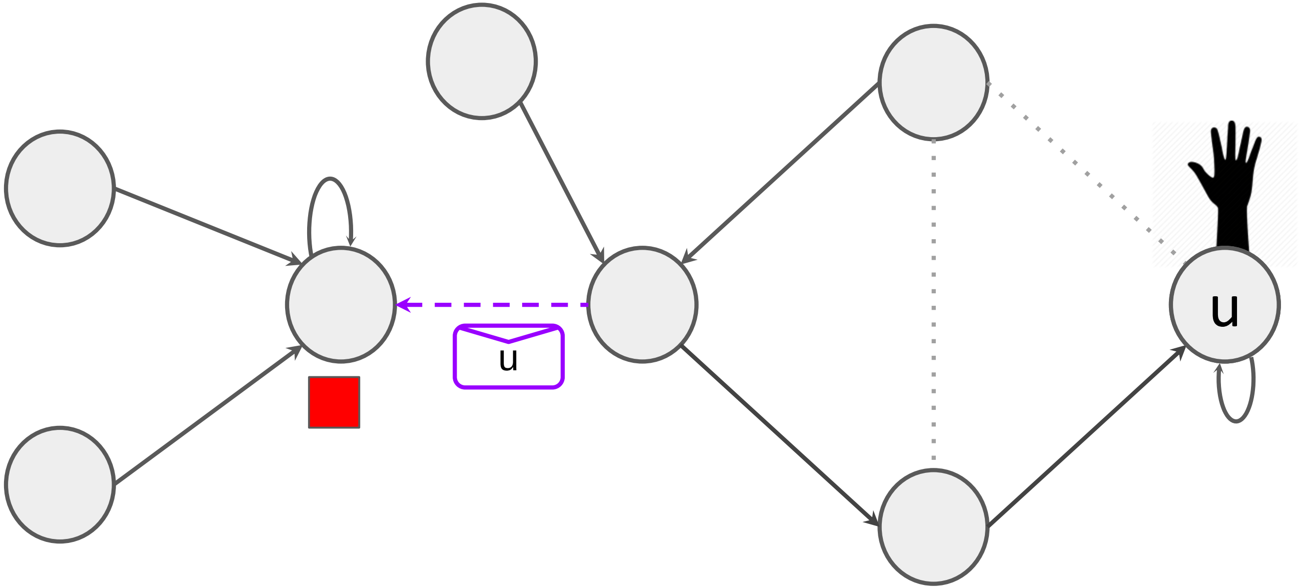
Arrow



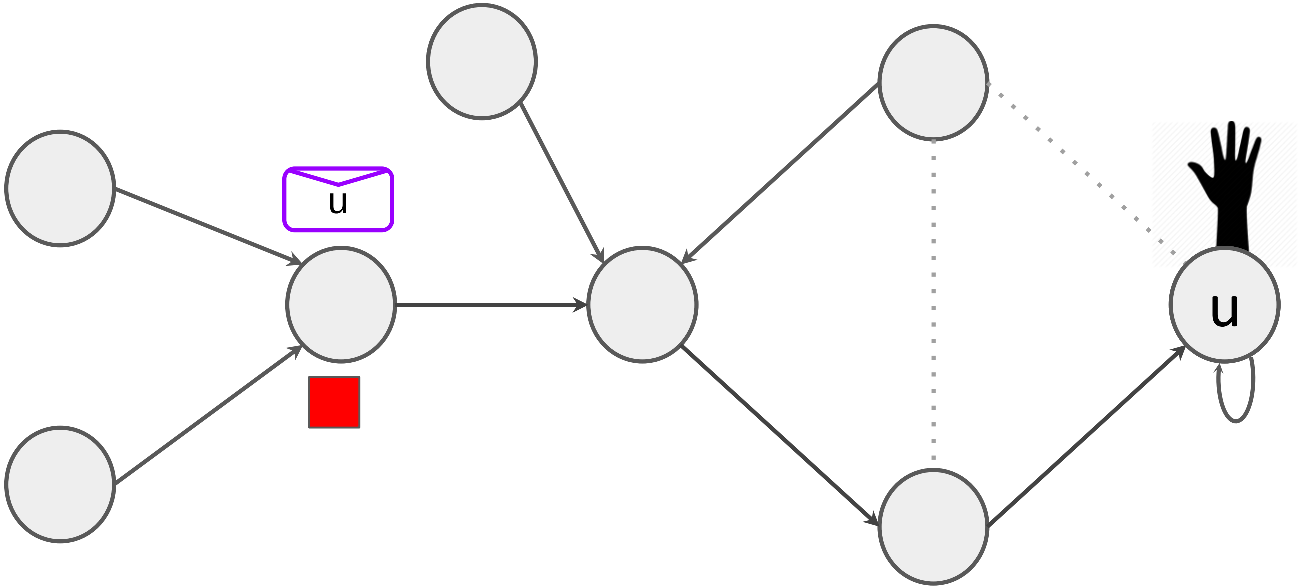
Arrow



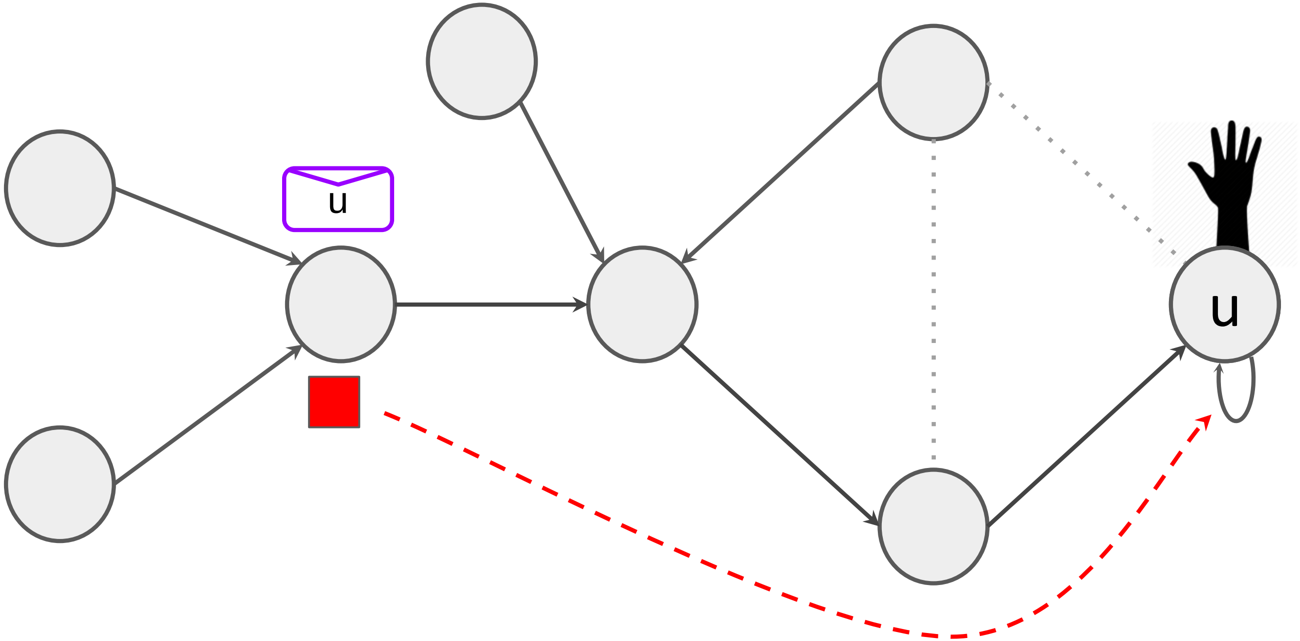
Arrow



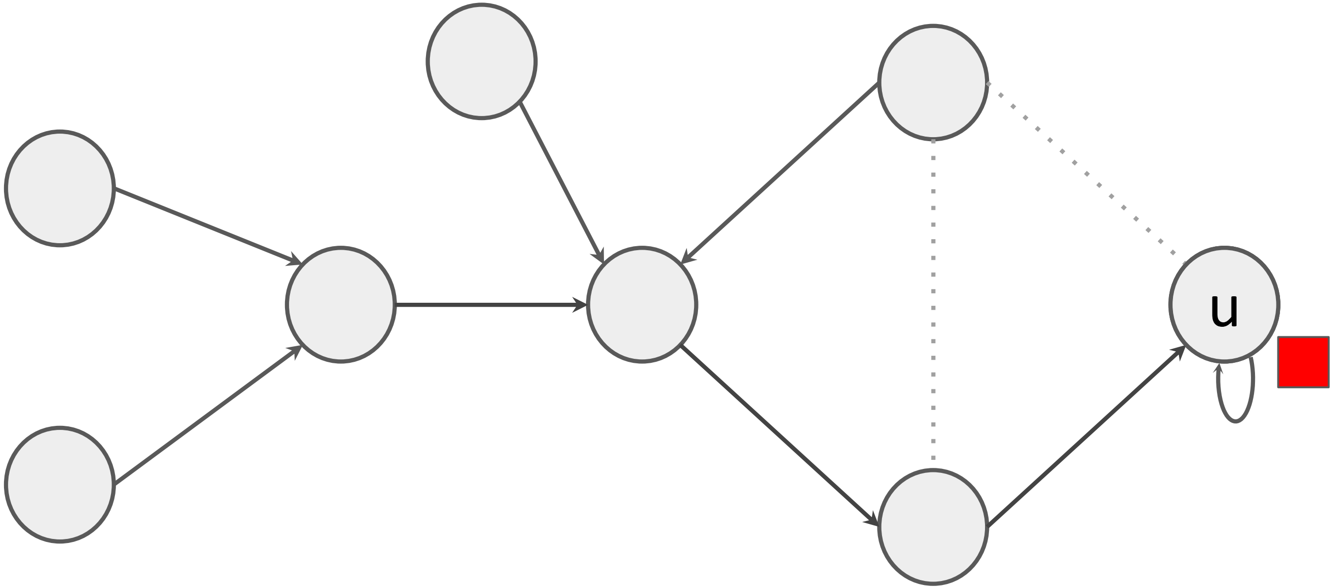
Arrow



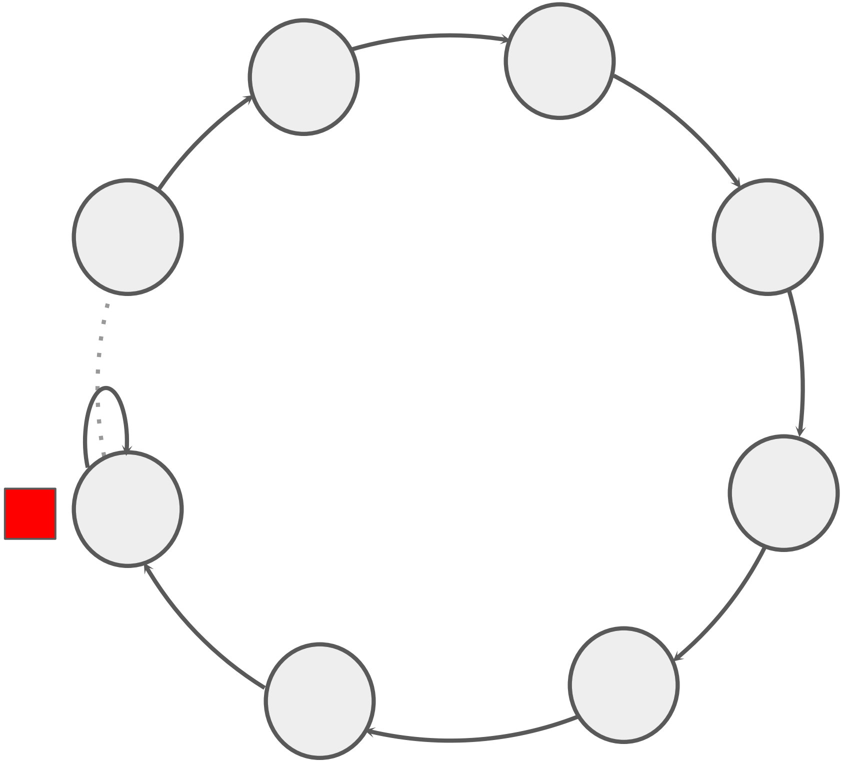
Arrow



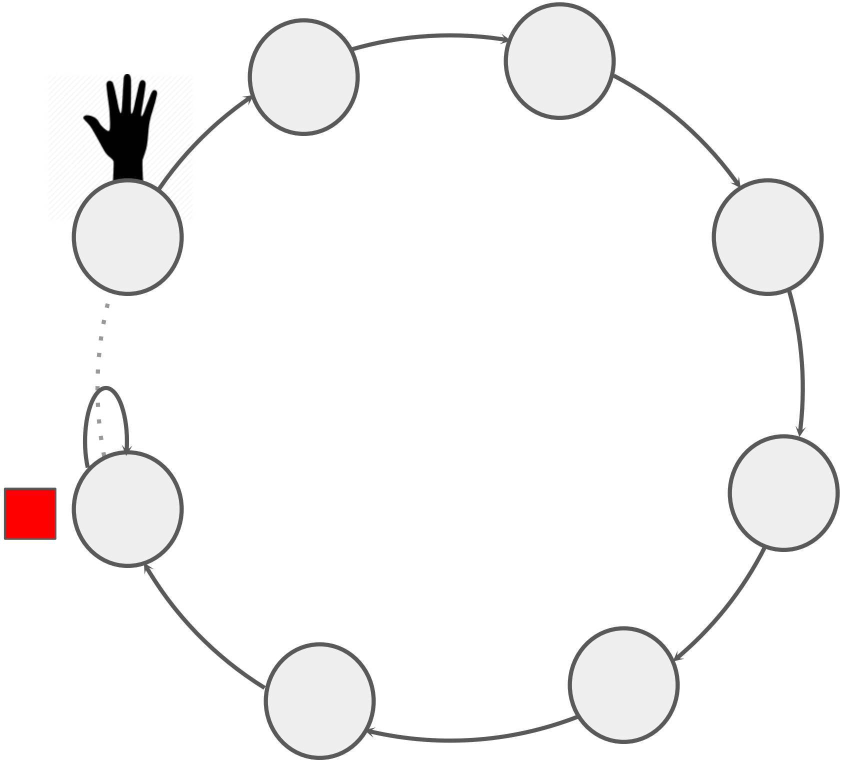
Arrow



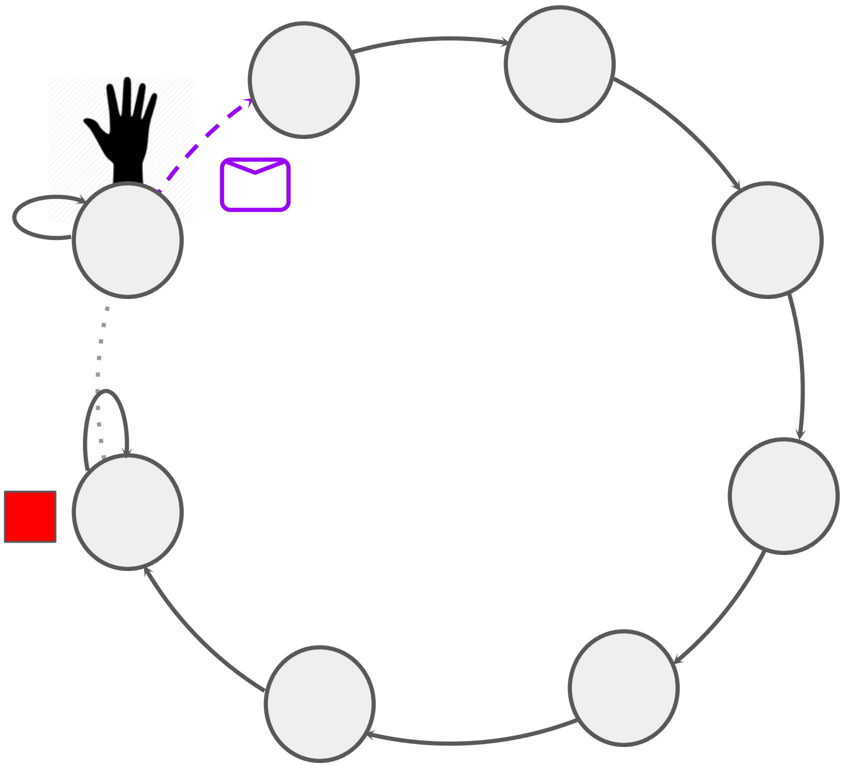
Arrow on Rings



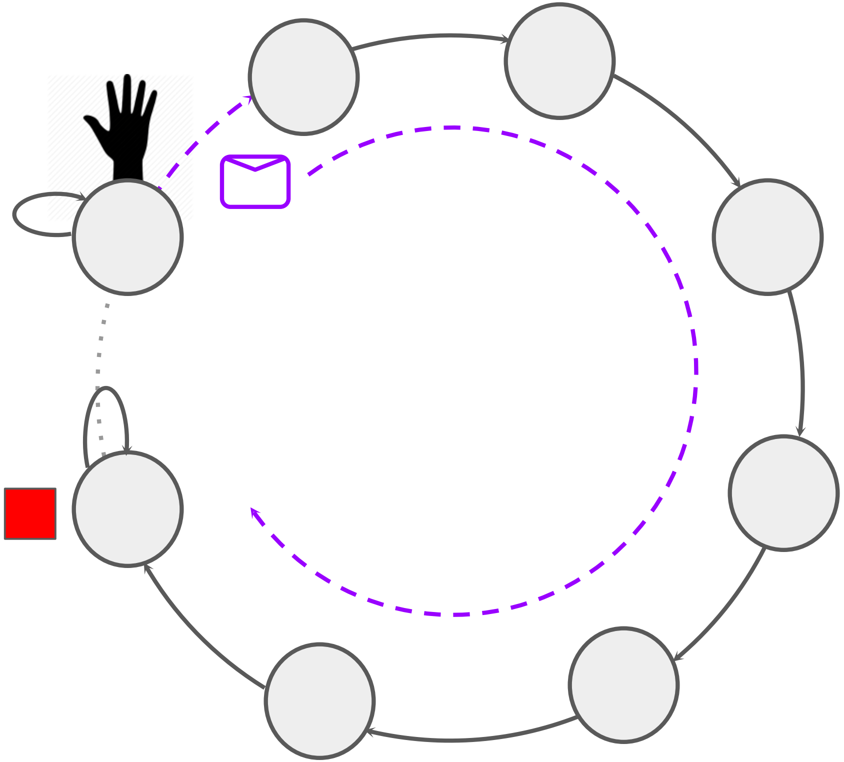
Arrow on Rings



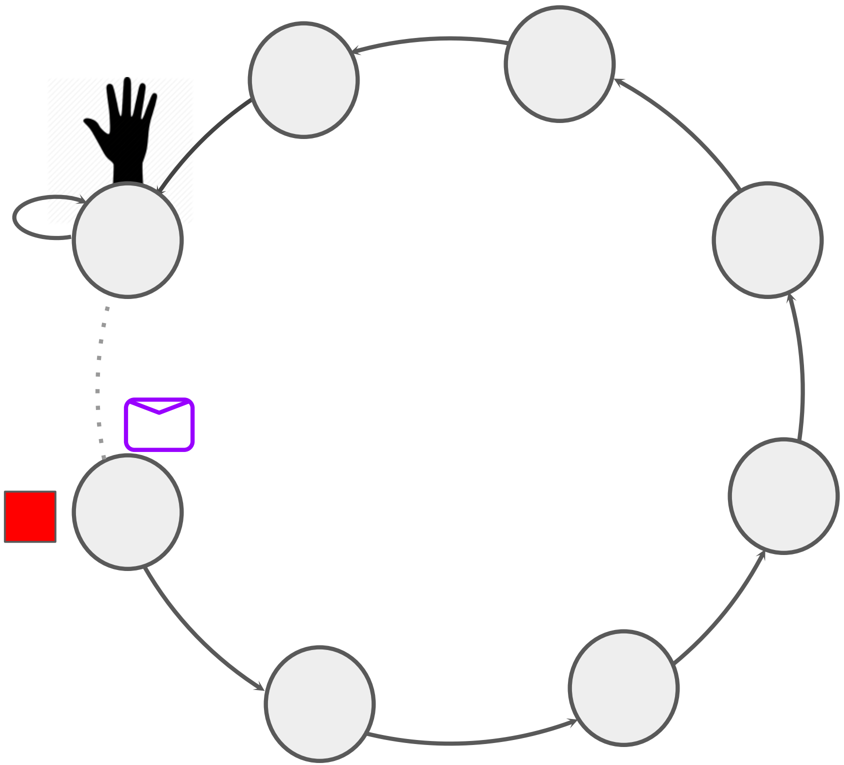
Arrow on Rings



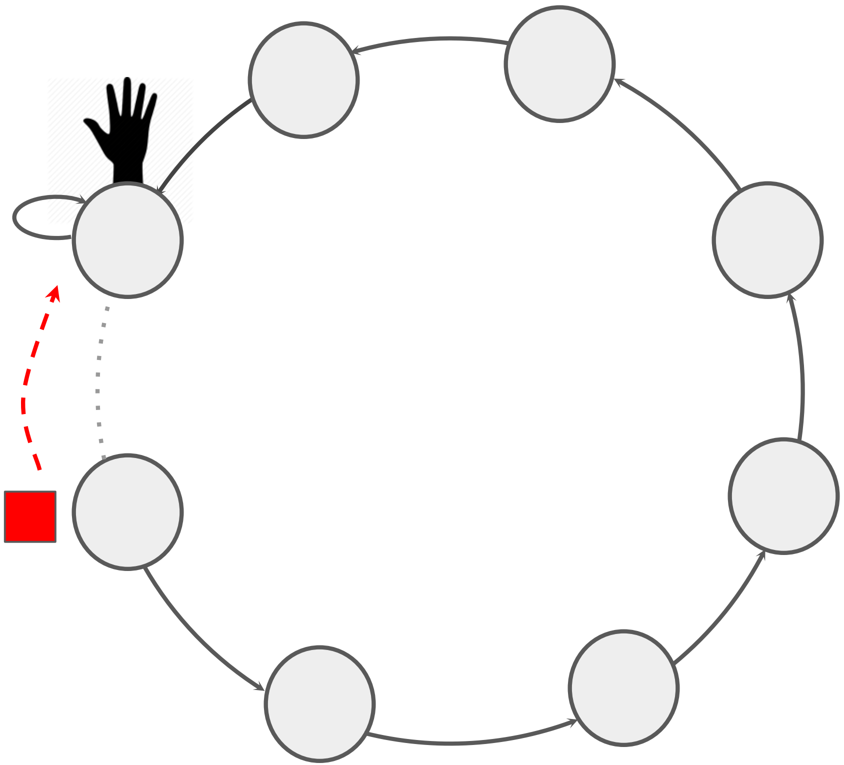
Arrow on Rings



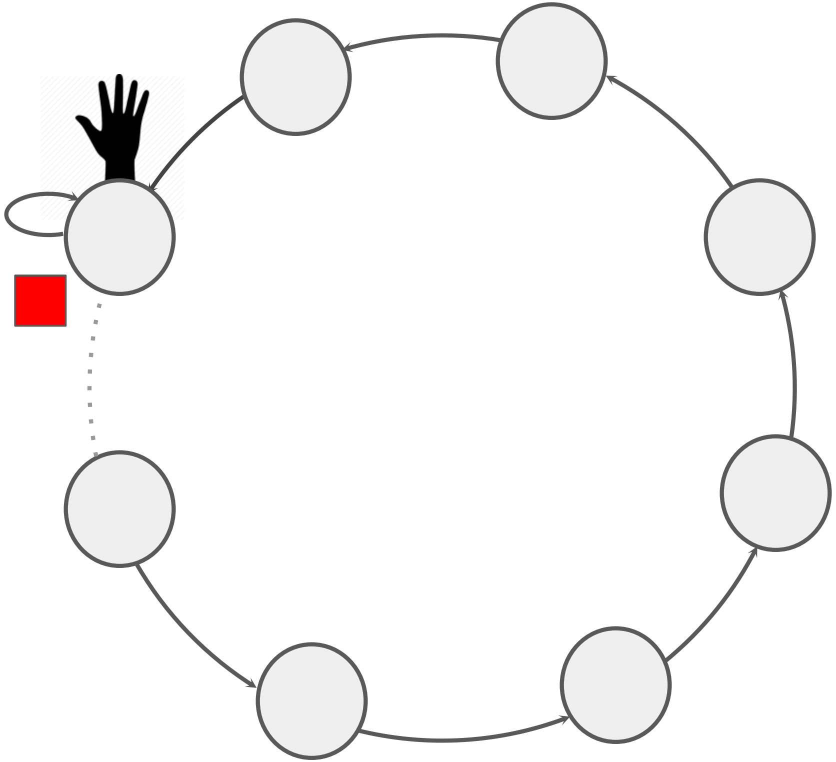
Arrow on Rings



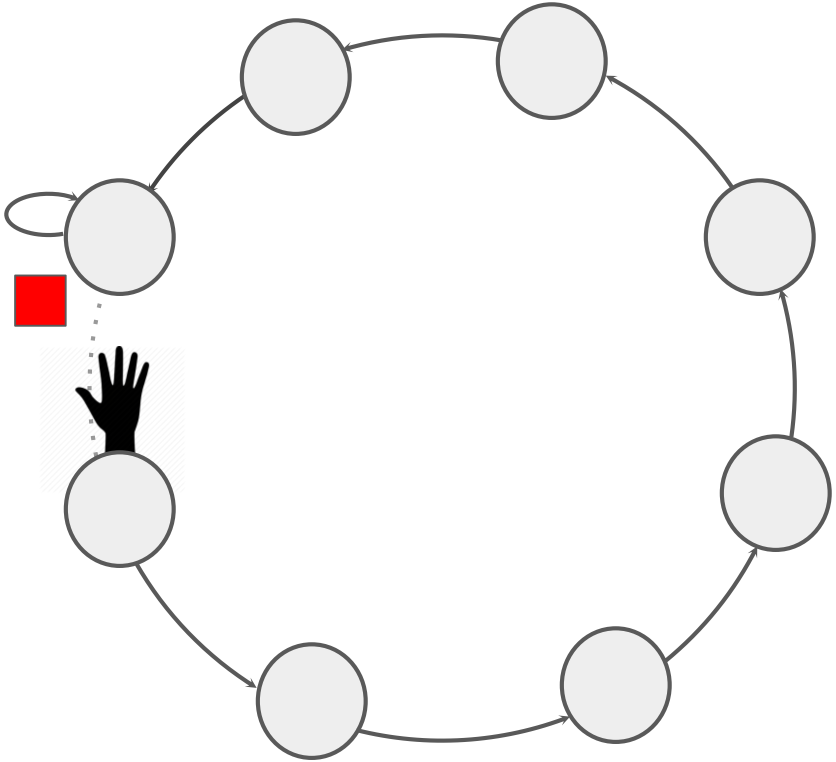
Arrow on Rings



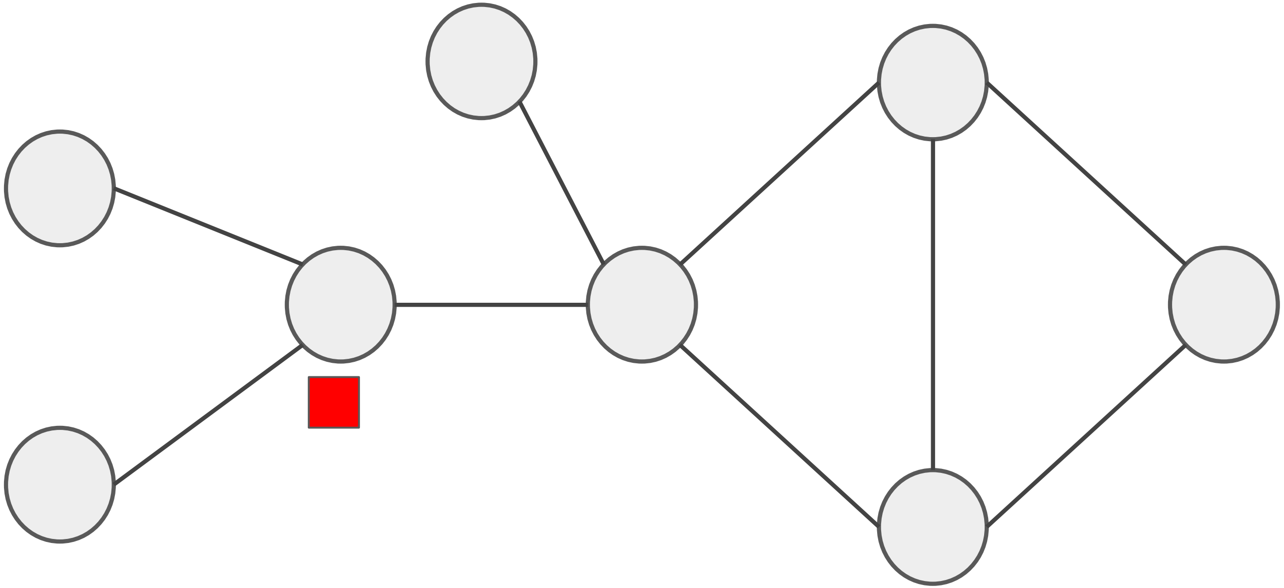
Arrow on Rings



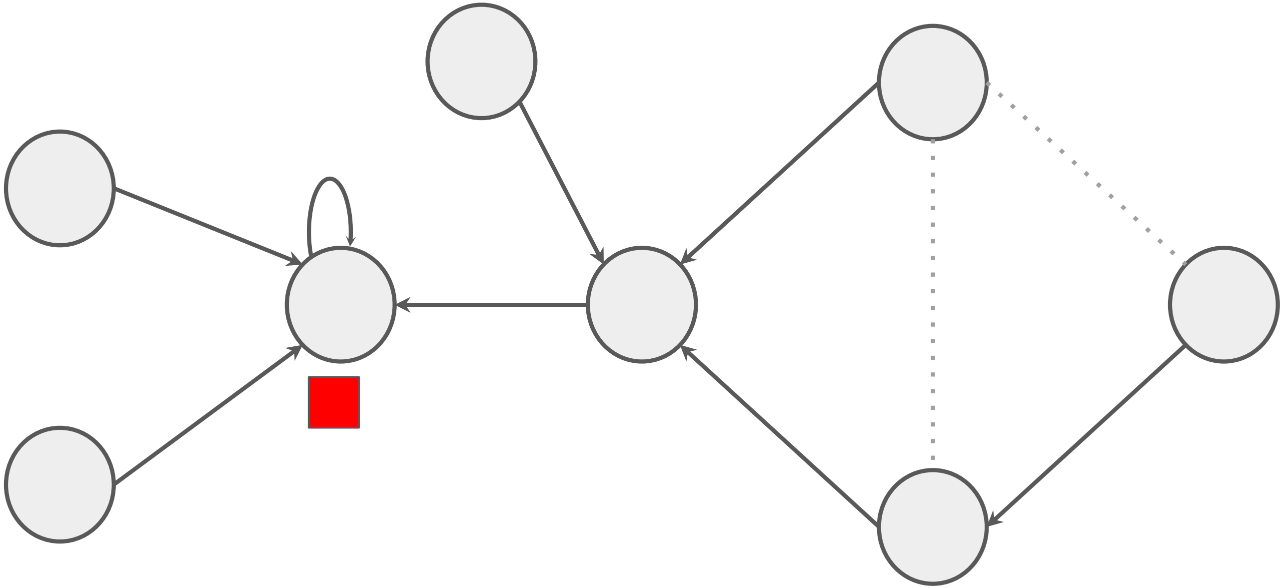
Arrow on Rings



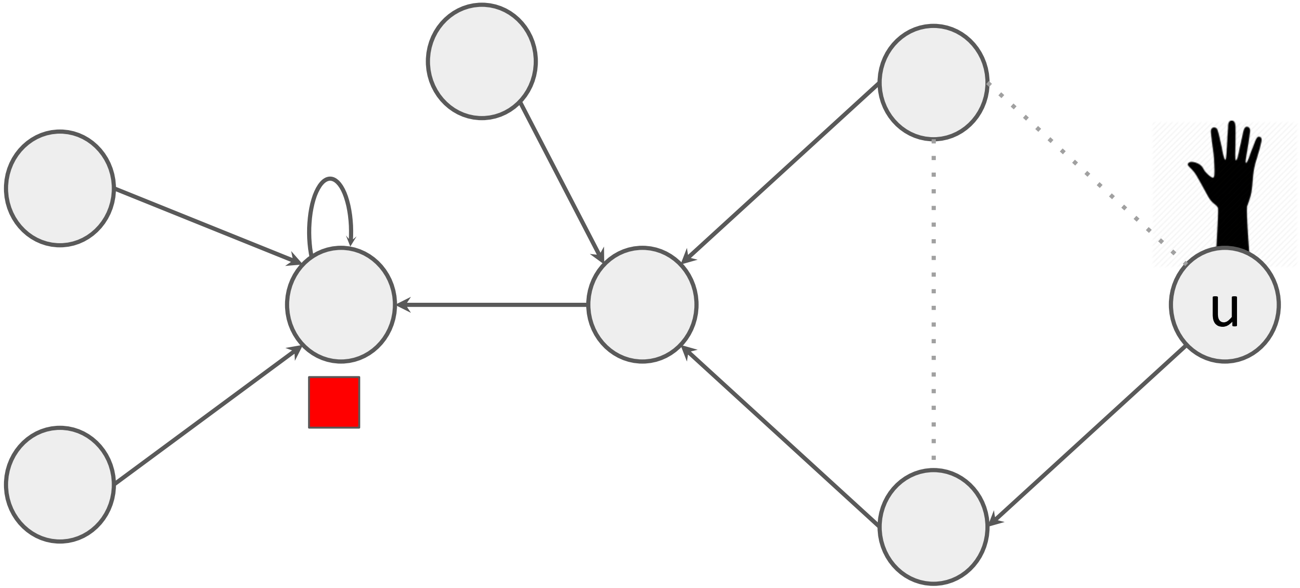
Arvy



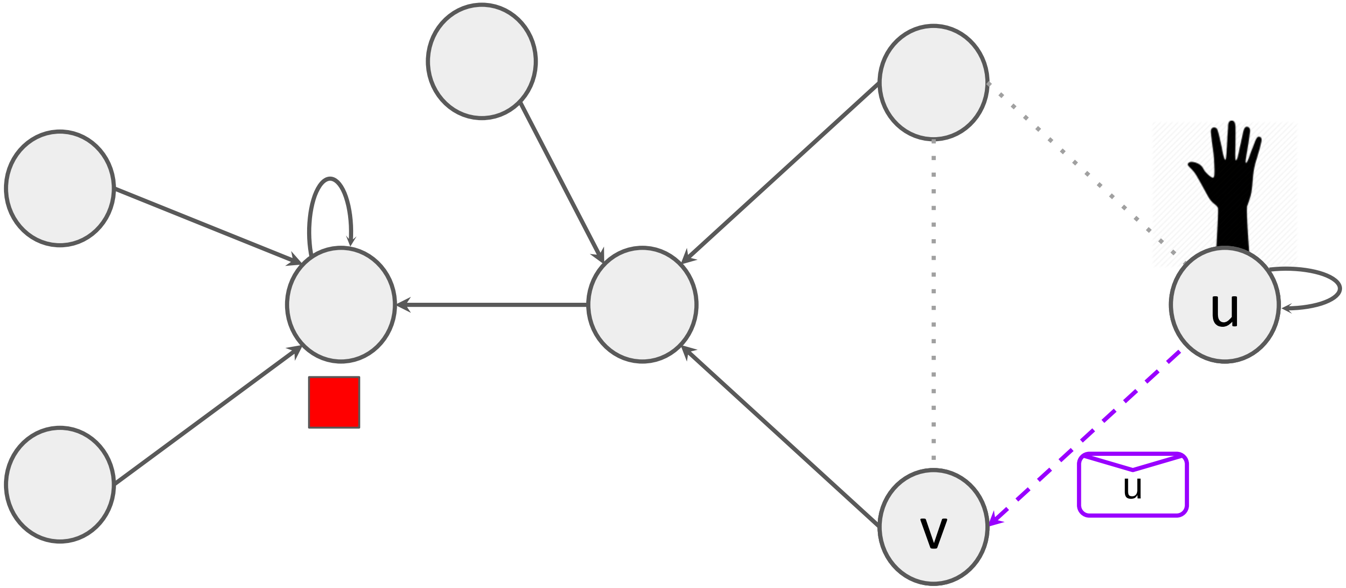
Arvy



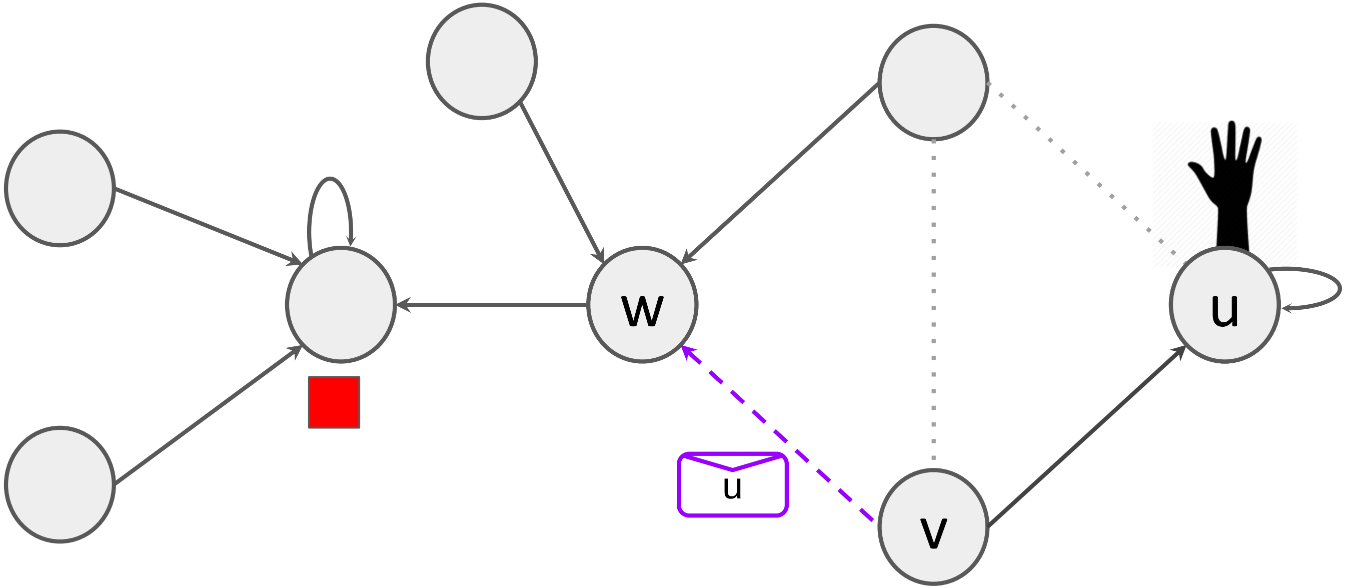
Arvy



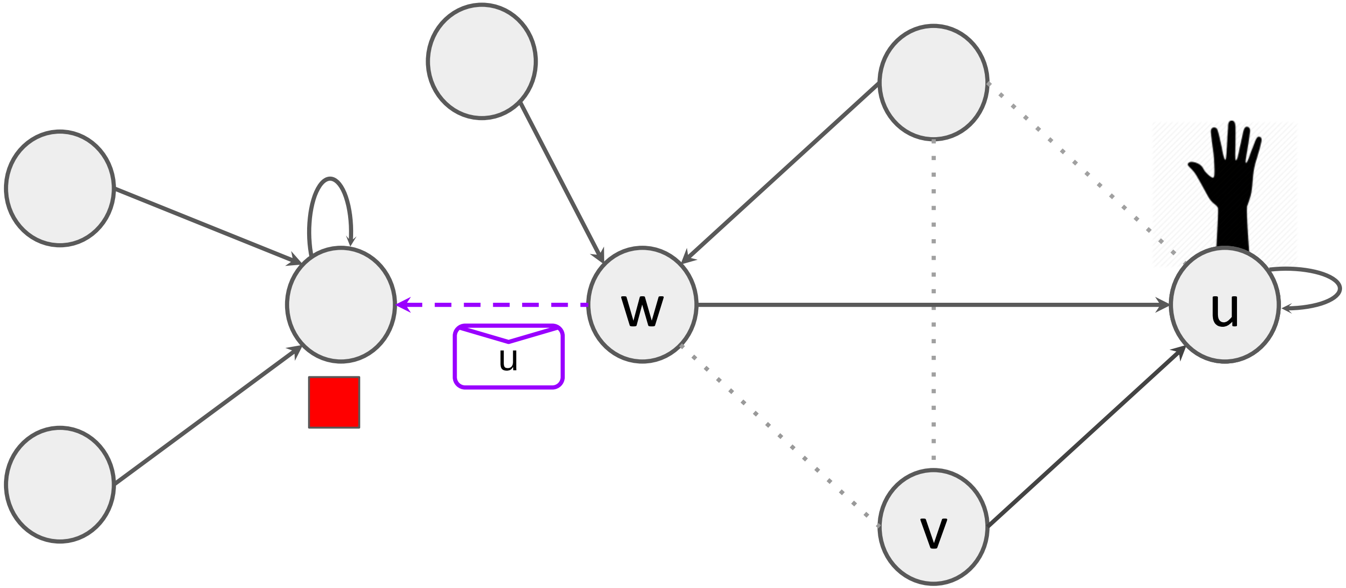
Arvy



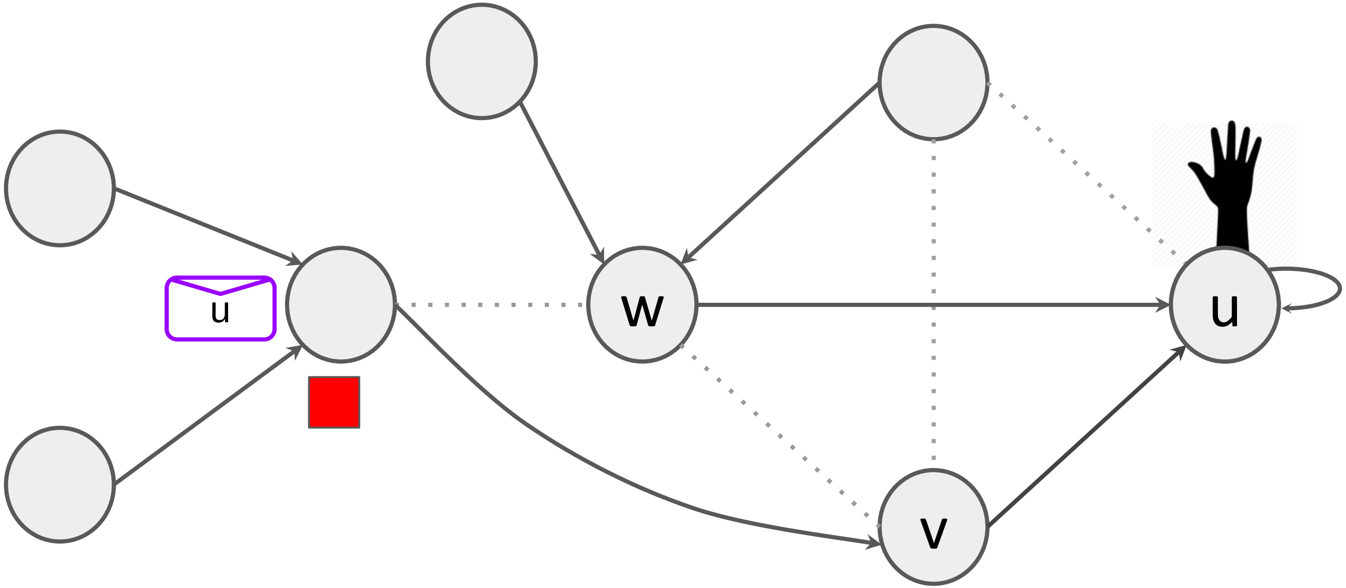
Arvy



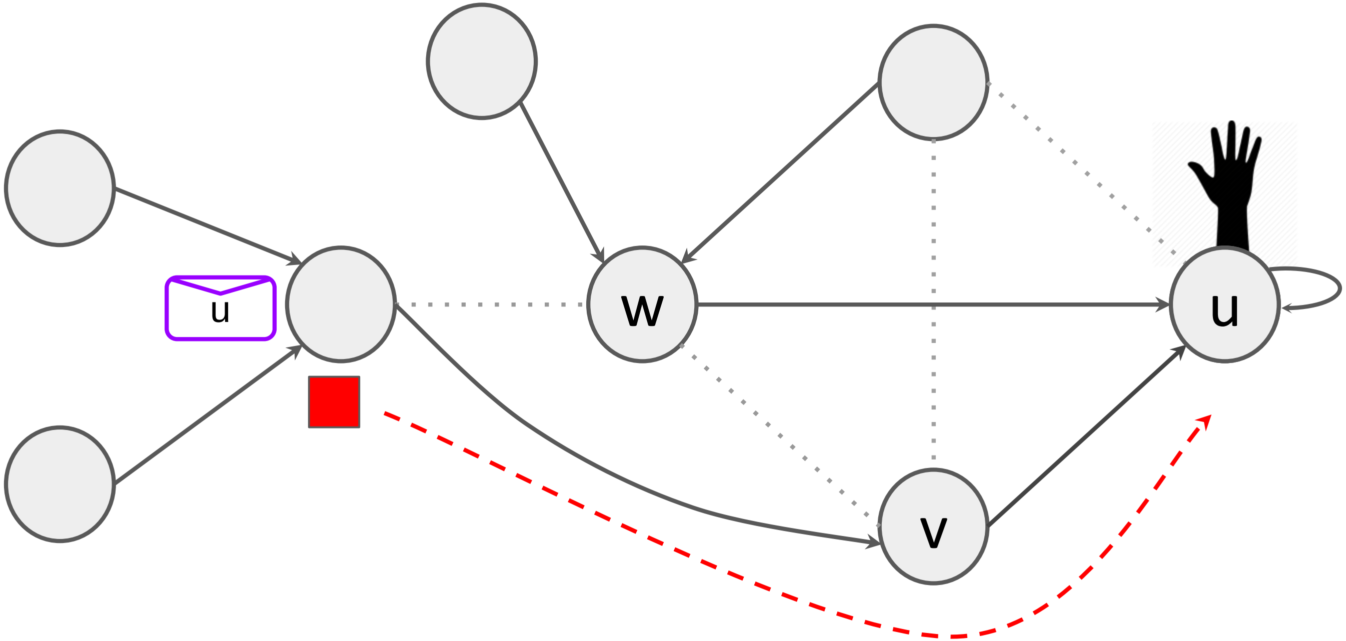
Arvy



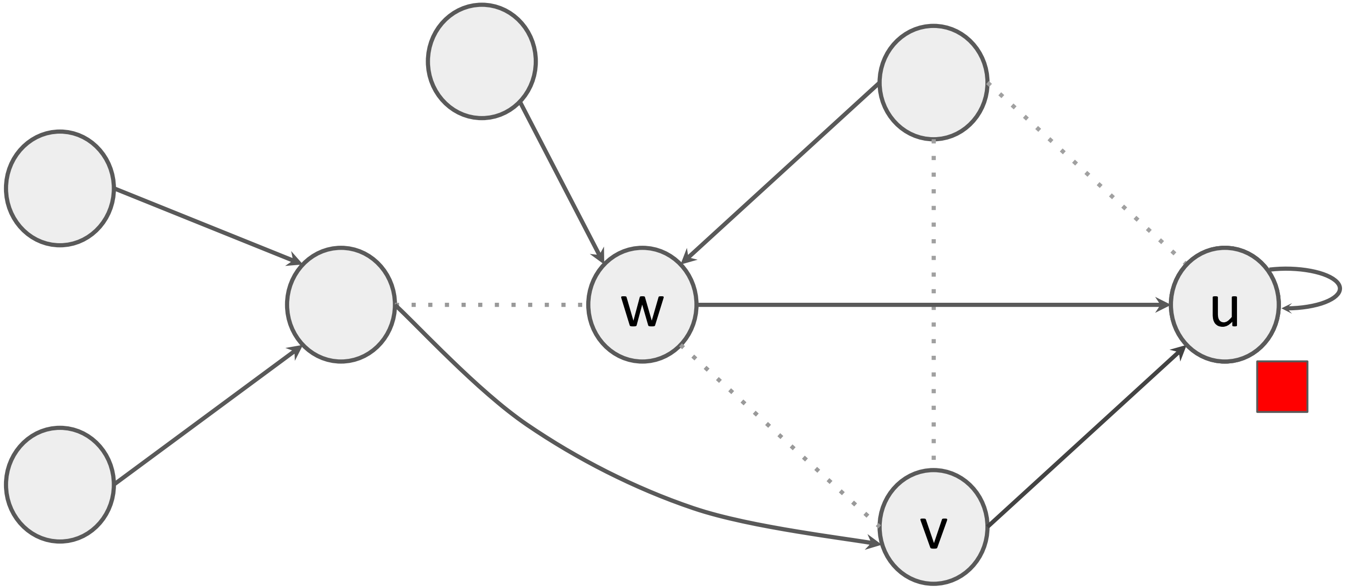
Arvy



Arvy

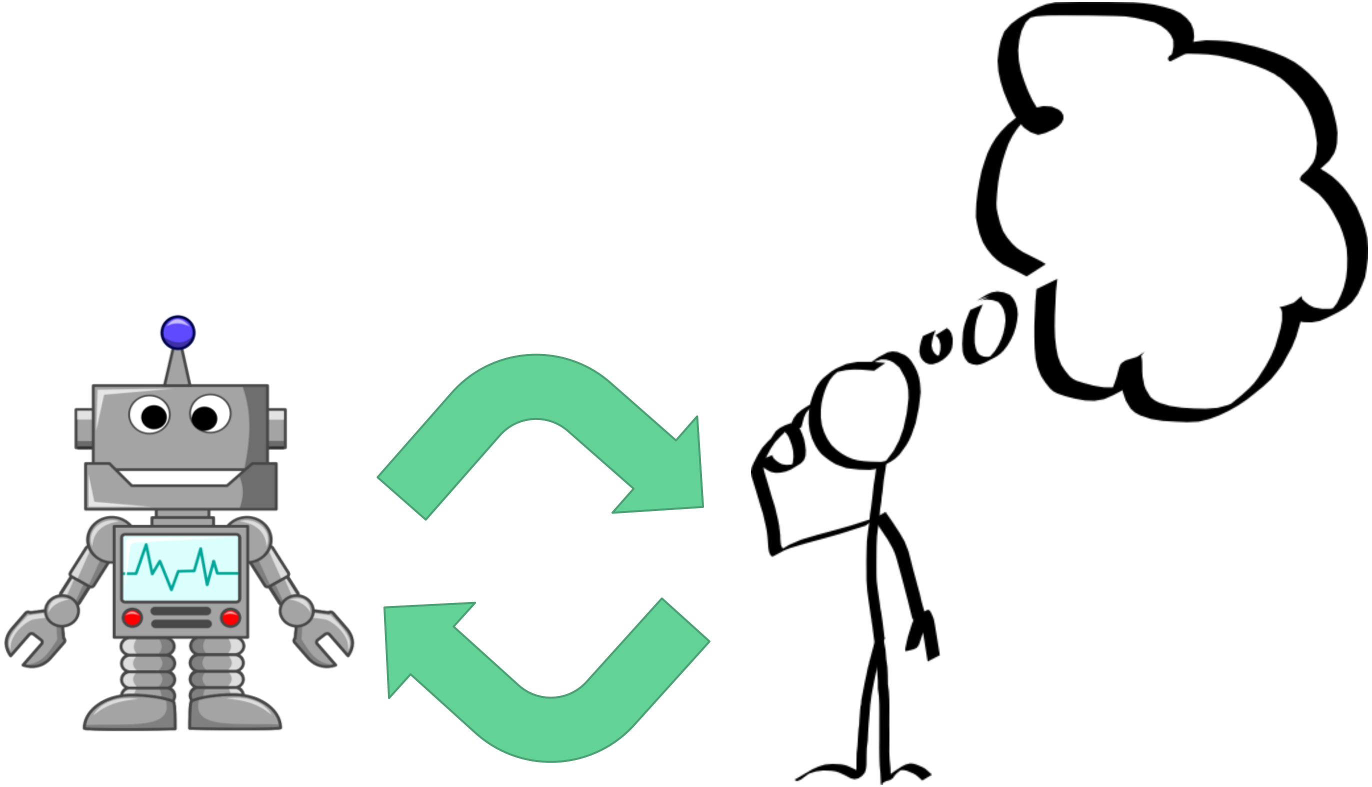


Arvy

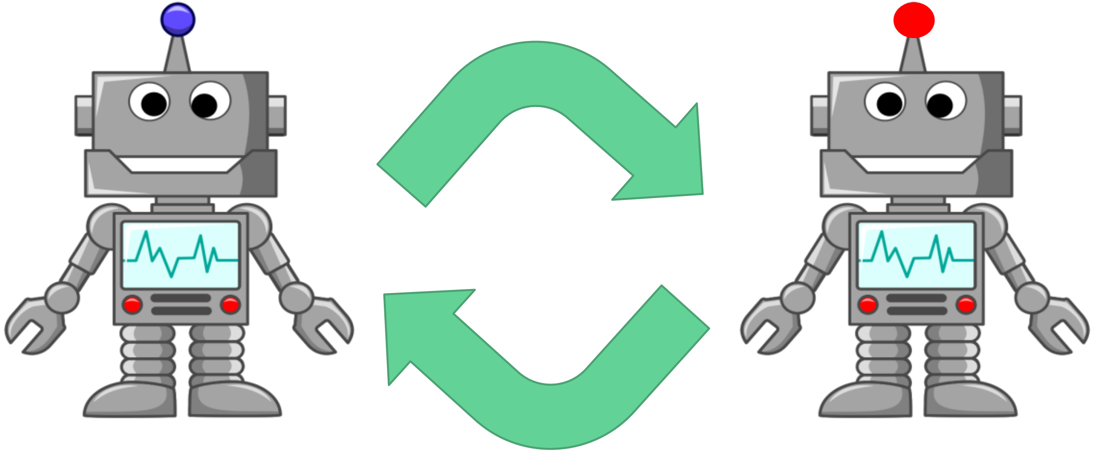


Performance

Graph	Protocol	Bound
Tree	Arrow	O(1)
Cycles	Arvy(Bridge)	O(1)
General?	Arvy(?)	?



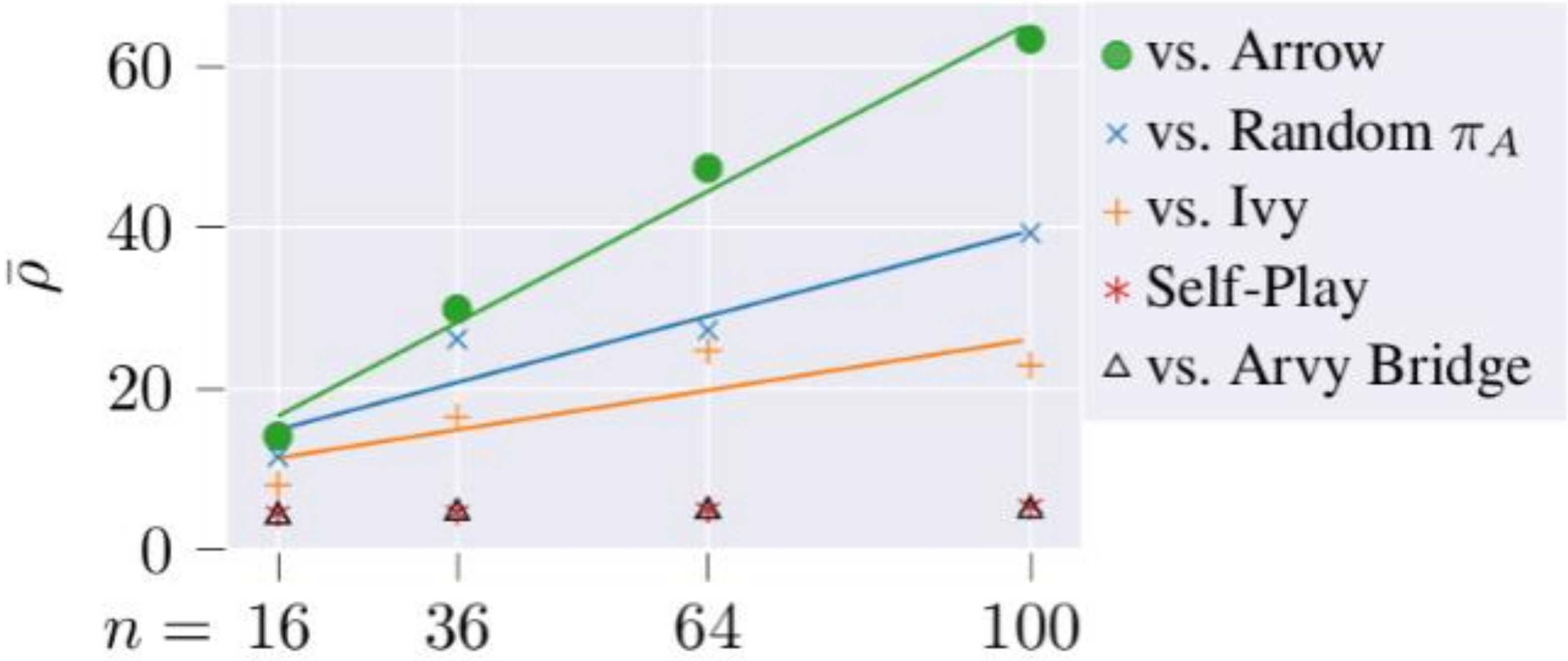
Arvy Agent $\boldsymbol{\pi}_A$ Request Agent $\boldsymbol{\pi}_{\boldsymbol{\sigma}}$



… trained to minimize competitive ratio

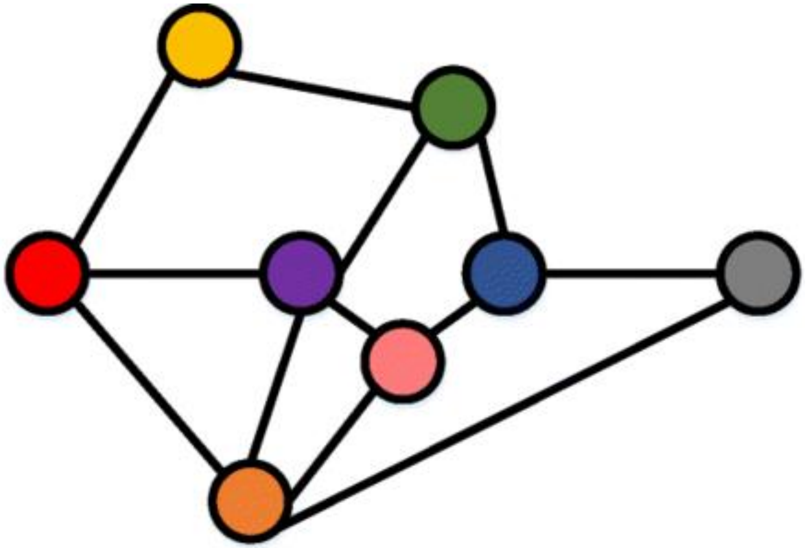
… trained to maximize competitive ratio

Results on Cycle



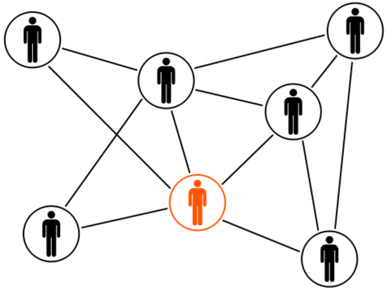

DC

Graph Neural Networks

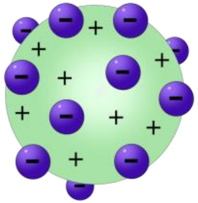


DeepMind

social networks

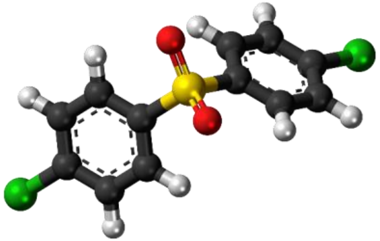


chemo-informatics



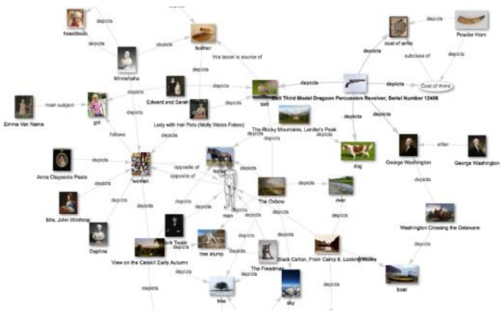
question answering systems

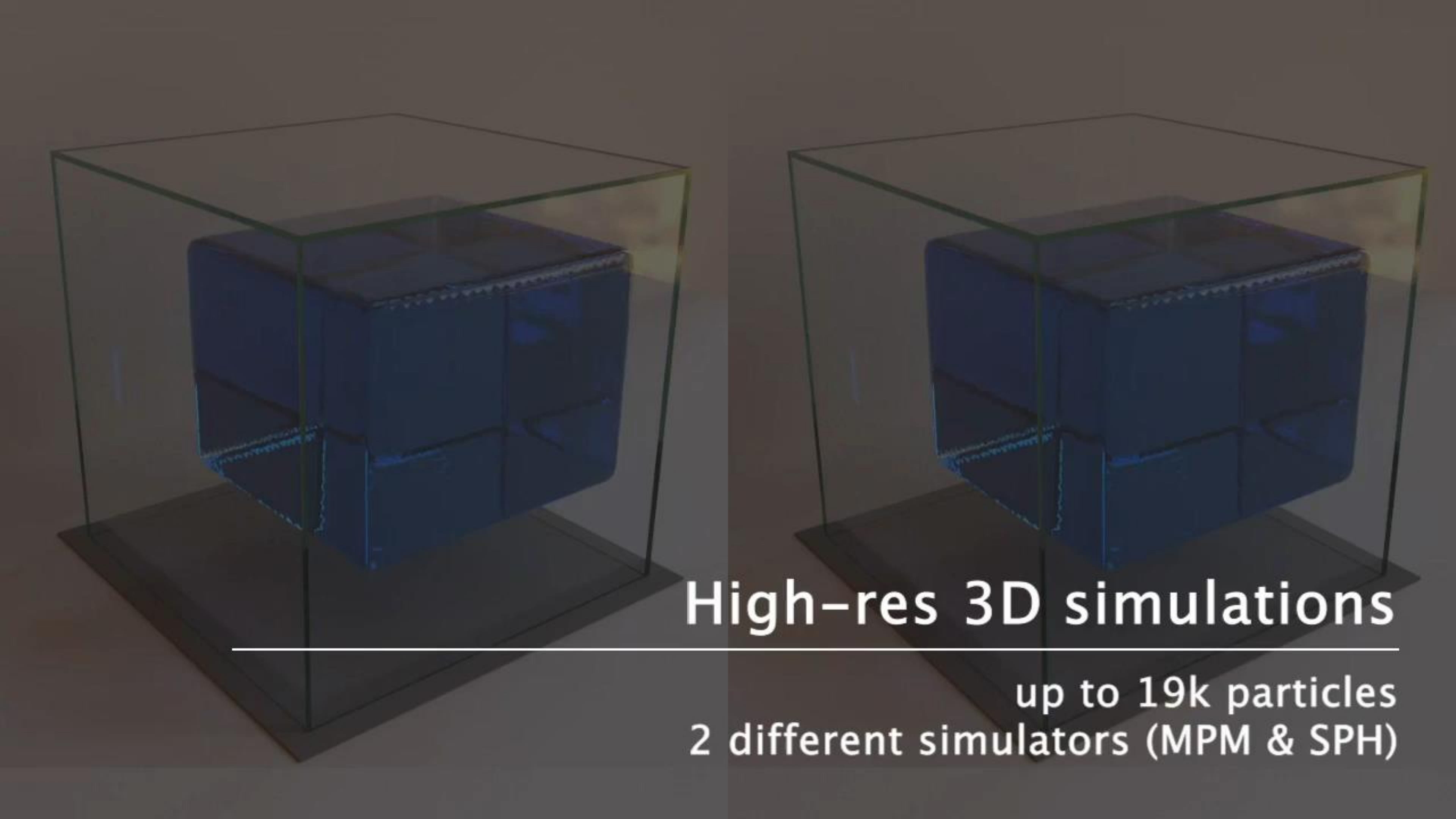
molecule recognition



recommender systems

knowledge graphs

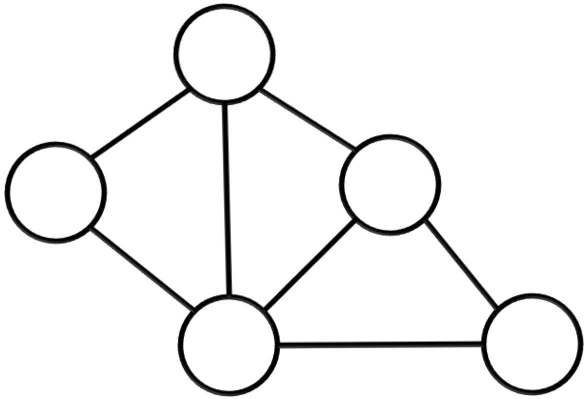



High-res 3D simulations
up to 19k particles
2 different simulators (MPM & SPH)

Distributed Computing (Message Passing)

Nodes communicate with neighbors by sending messages.

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

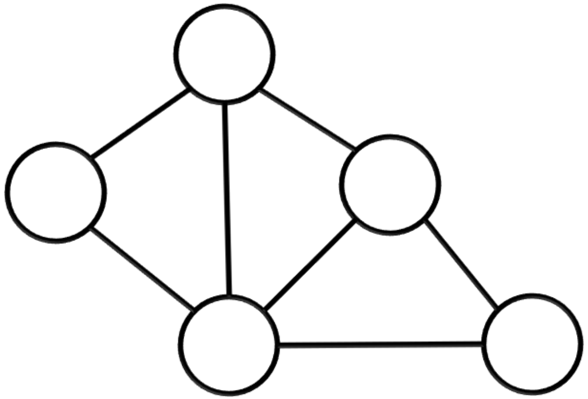


```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```

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

ML Track

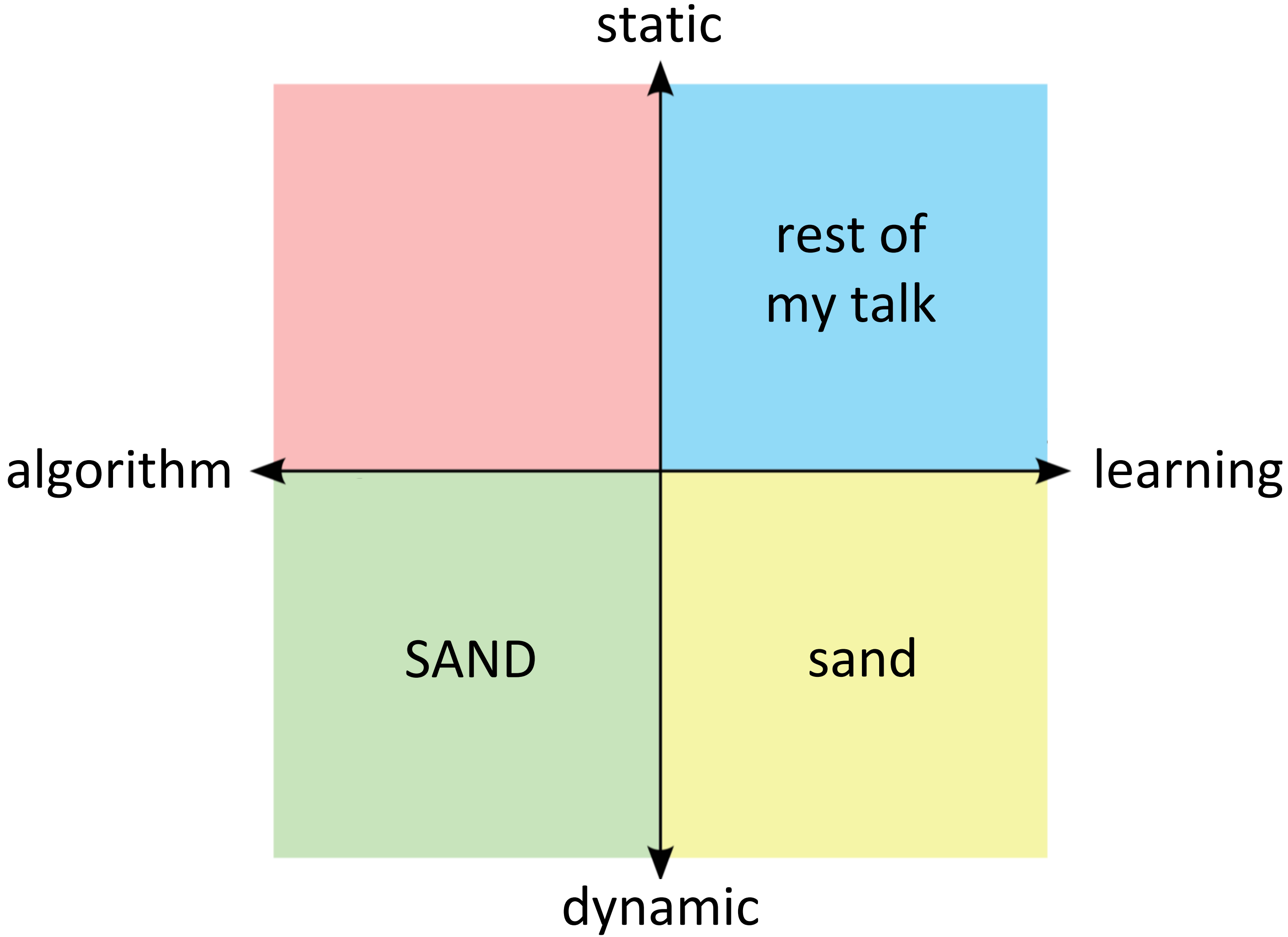
"Learned" parameters

Usually node features

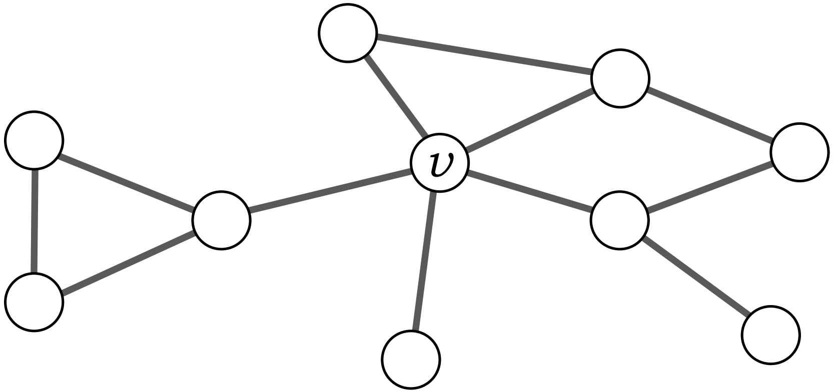
Aggregated messages

Solve classification (node, edge, graph)

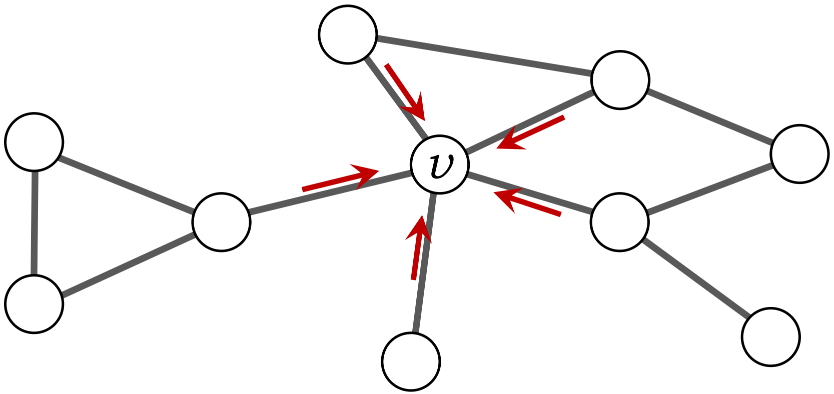
```
each round:
every node:
1. send msgs
2. rcv msgs
3. compute
```


static
rest of
my talk
algorithm
learning
SAND
sand
dynamic

Graph Neural Networks

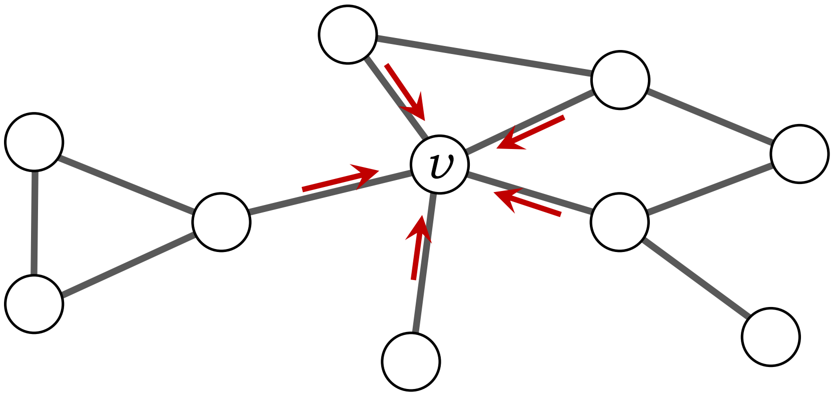


Graph Neural Networks



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

Graph Neural Networks

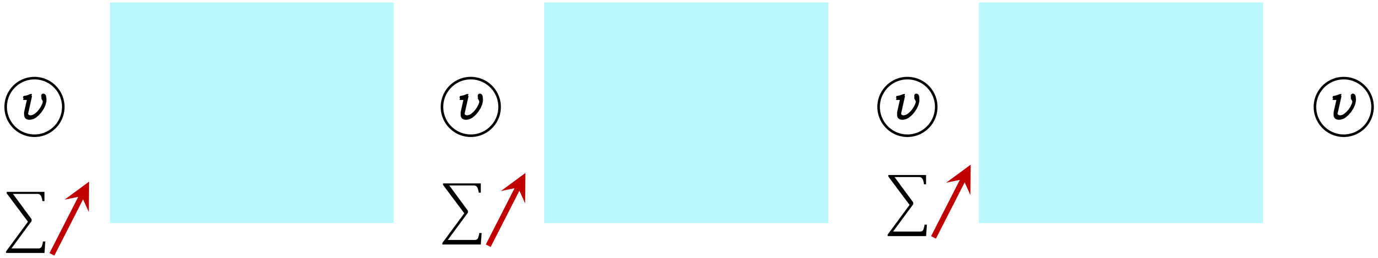


$$a_v = \textsc{Aggregate}\left(\{\{\, h_u \mid u \in N(v) \,\}\}\right)$$

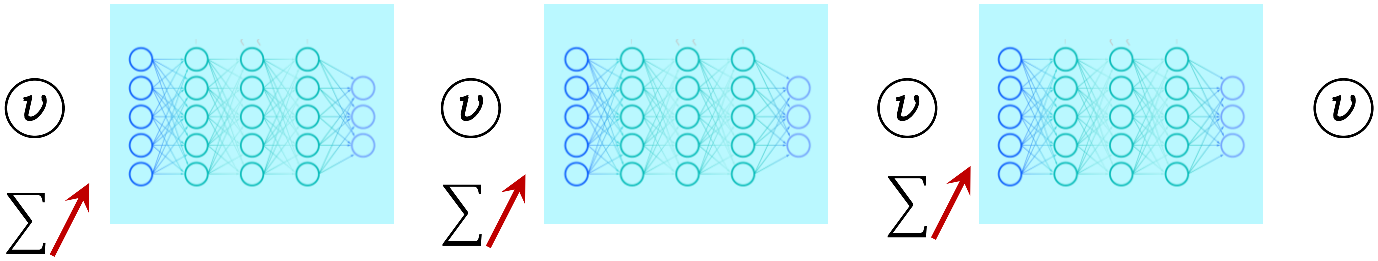
$$h_v^{(t+1)} = \textsc{Update}\left(h_v, a_v\right)$$

Graph Neural Networks

Graph Neural Networks



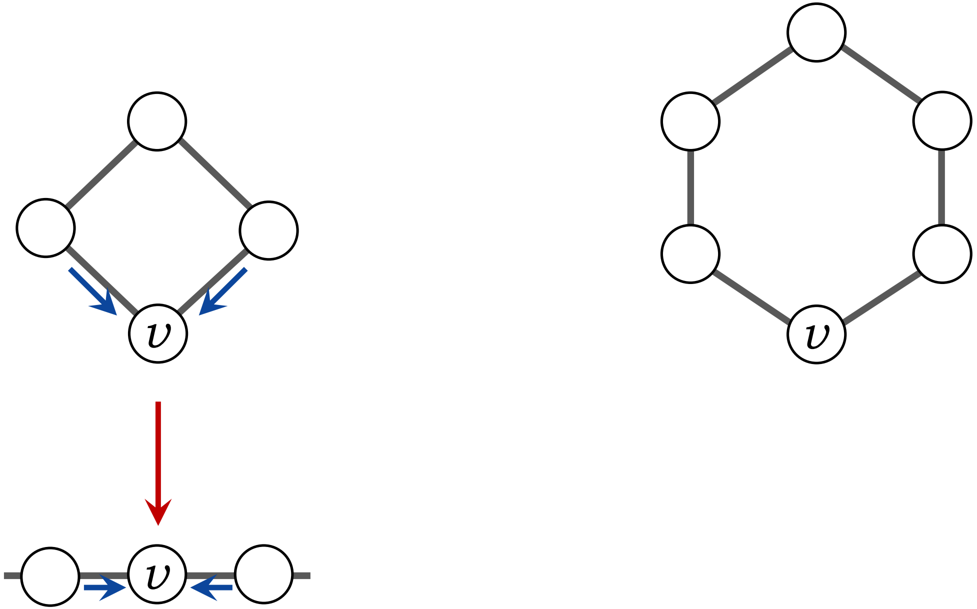
Graph Neural Networks



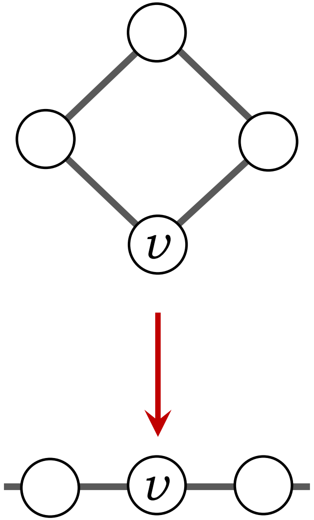
Limitations of GNNs?

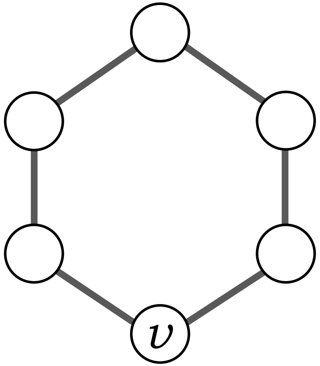
Limits of GNNs

Limits of GNNs

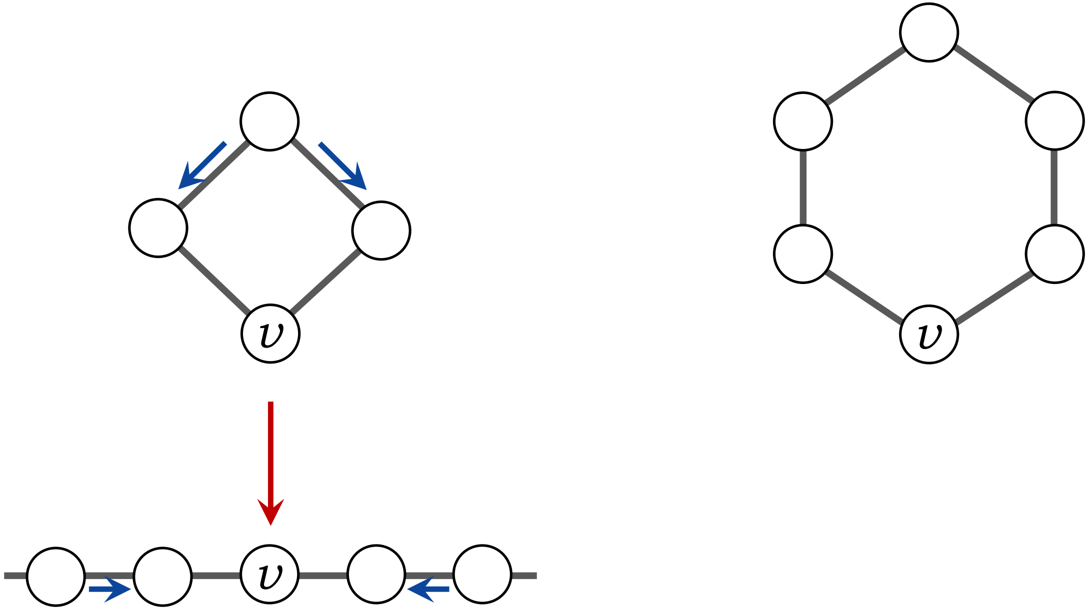


Limits of GNNs

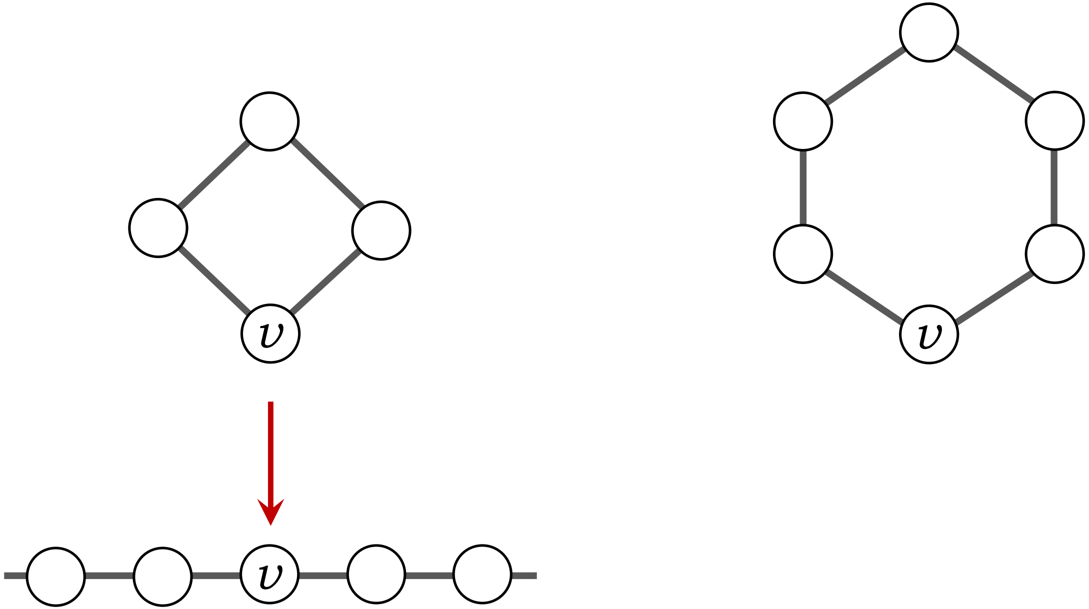




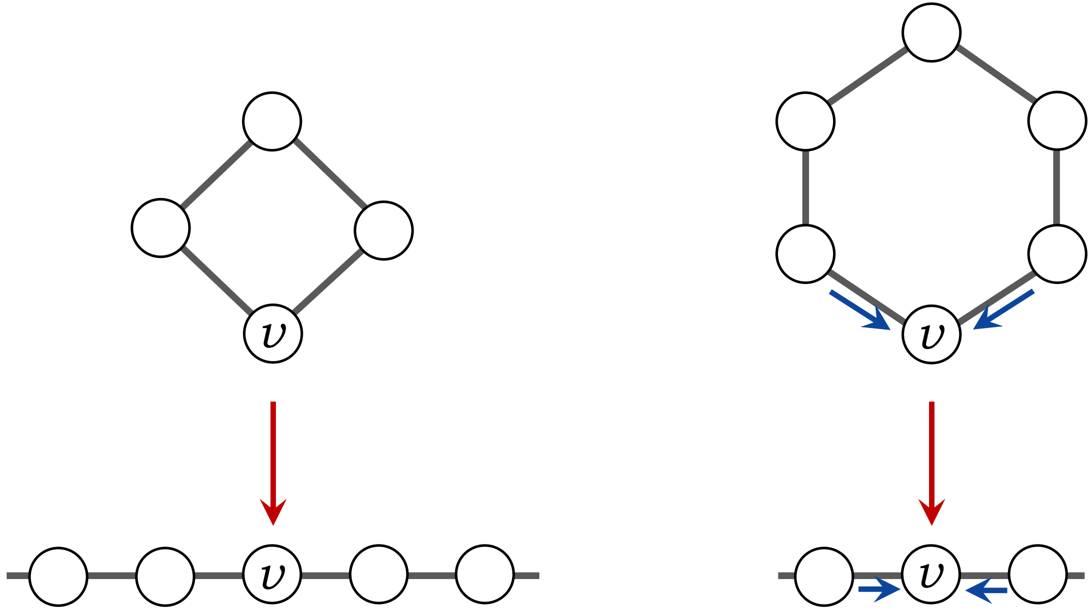
Limits of GNNs



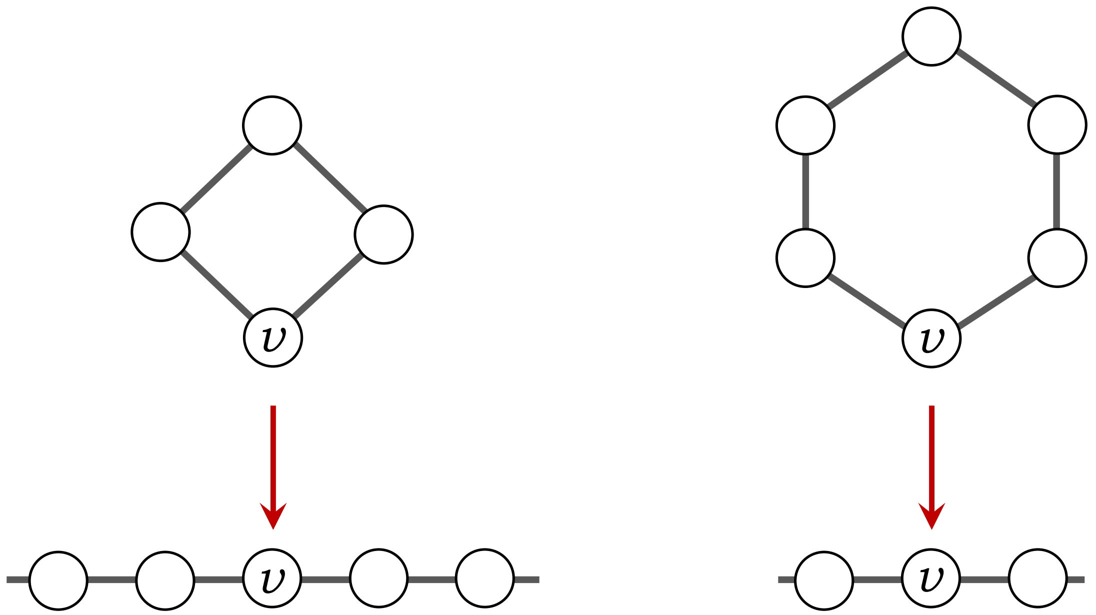
Limits of GNNs



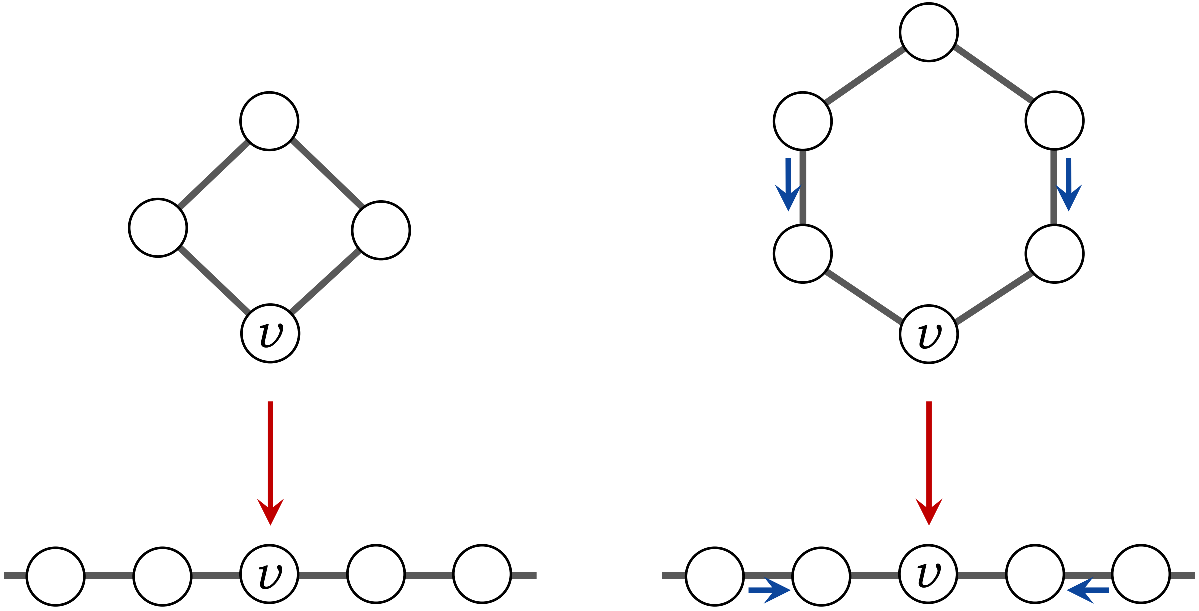
Limits of GNNs



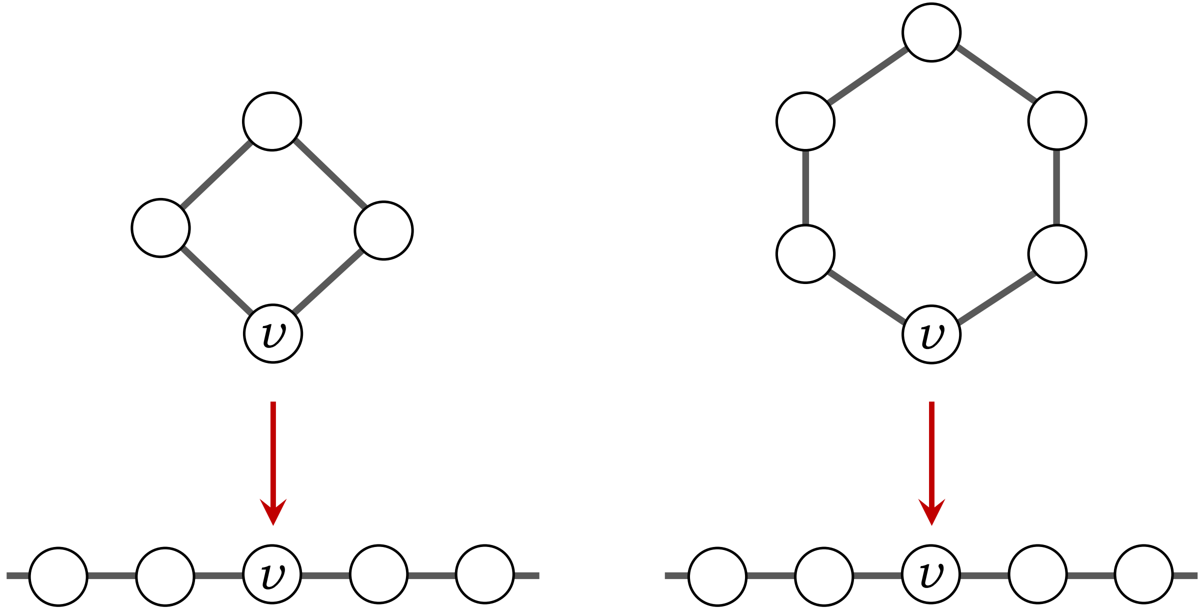
Limits of GNNs



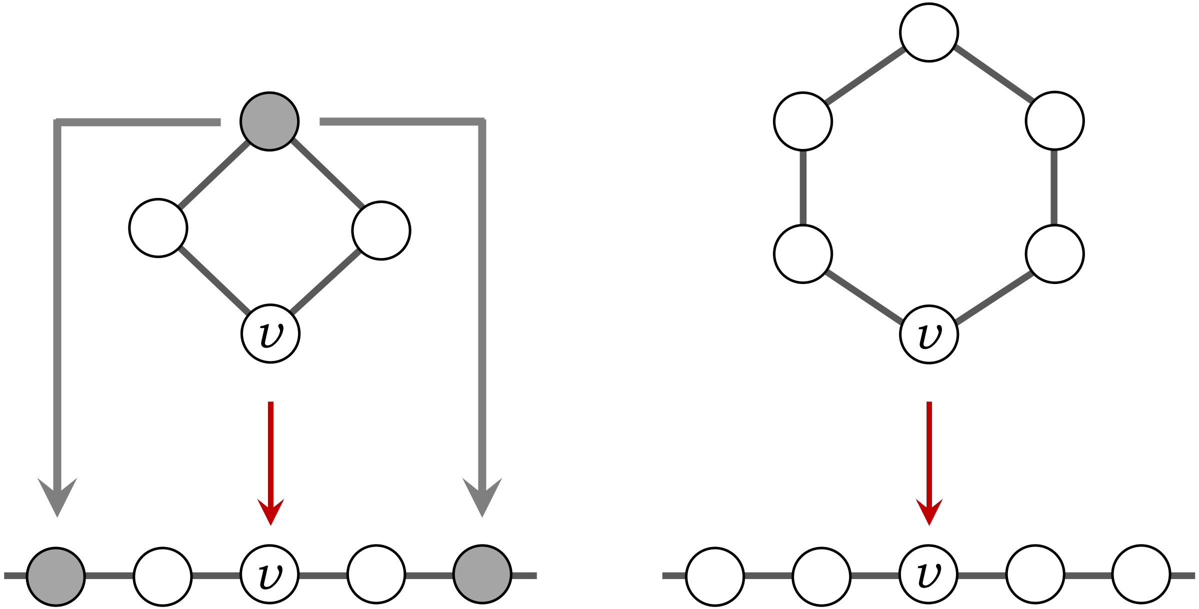
Limits of GNNs



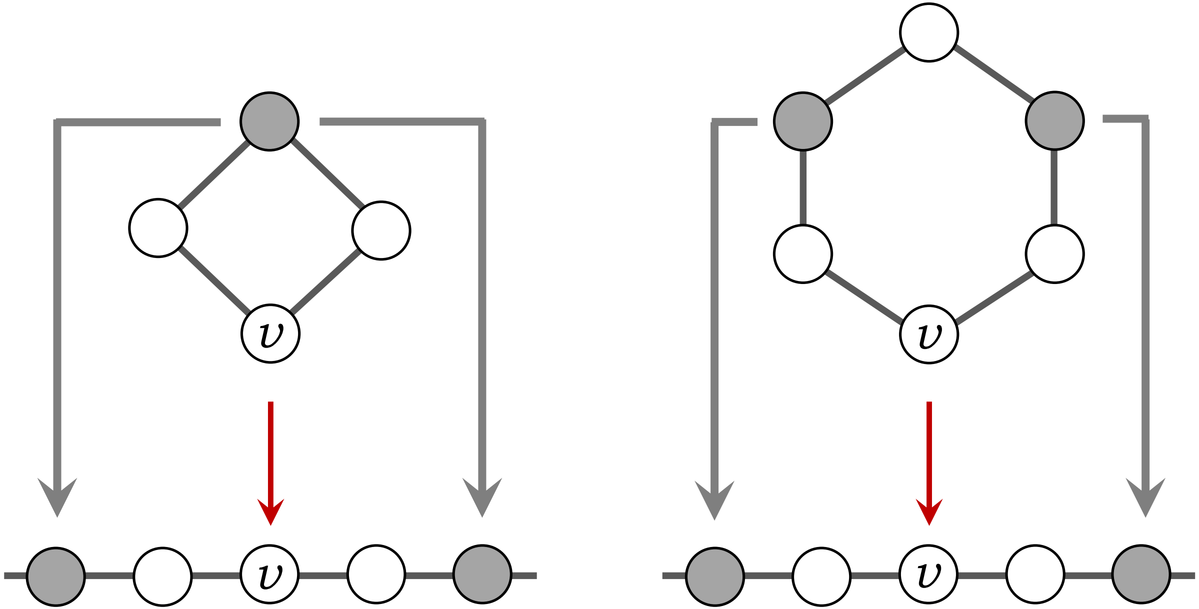
Limits of GNNs



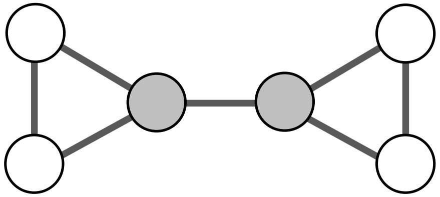
Limits of GNNs

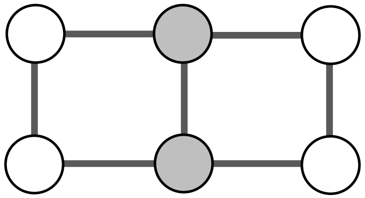


Limits of GNNs

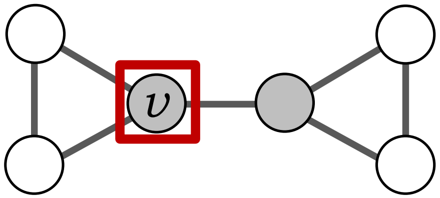


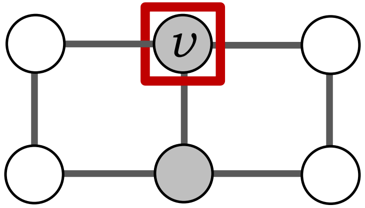
Limits of GNNs



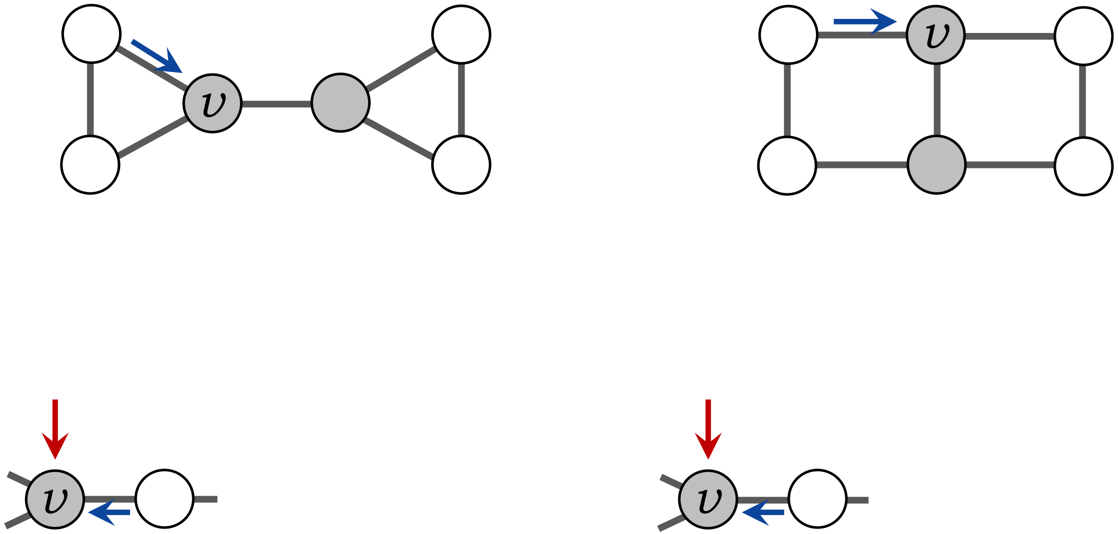


Limits of GNNs

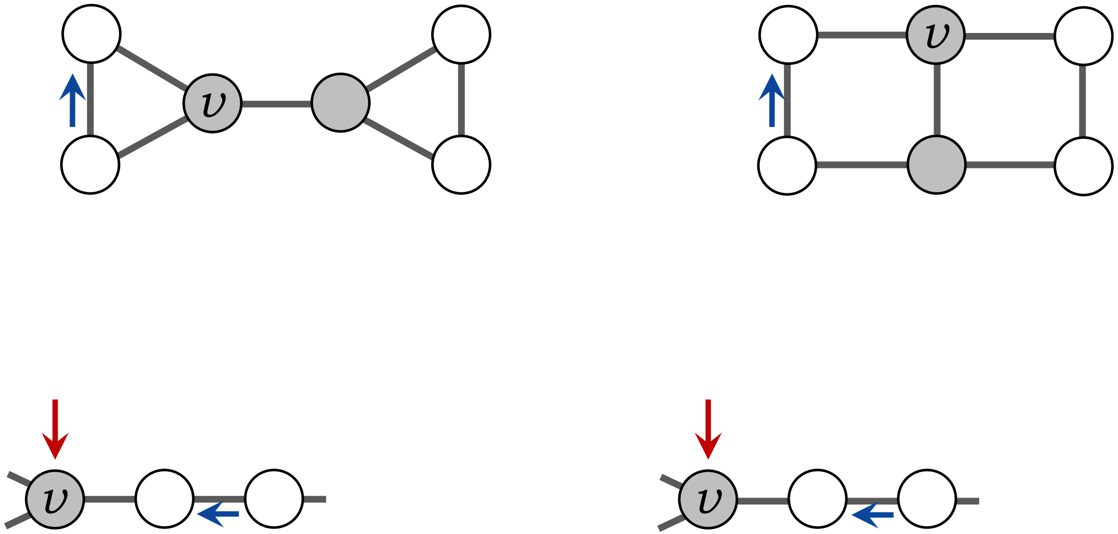




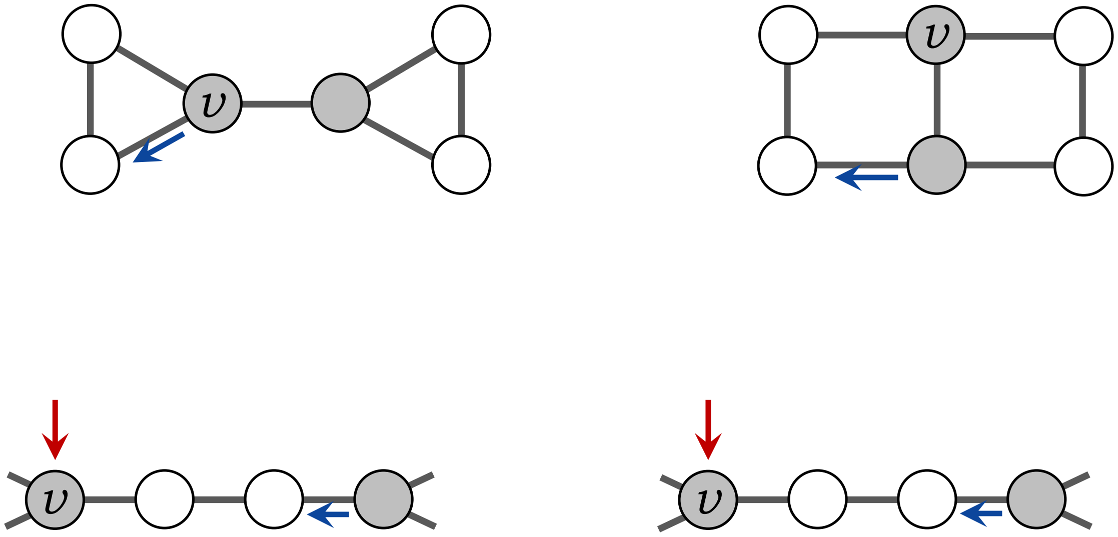
Limits of GNNs



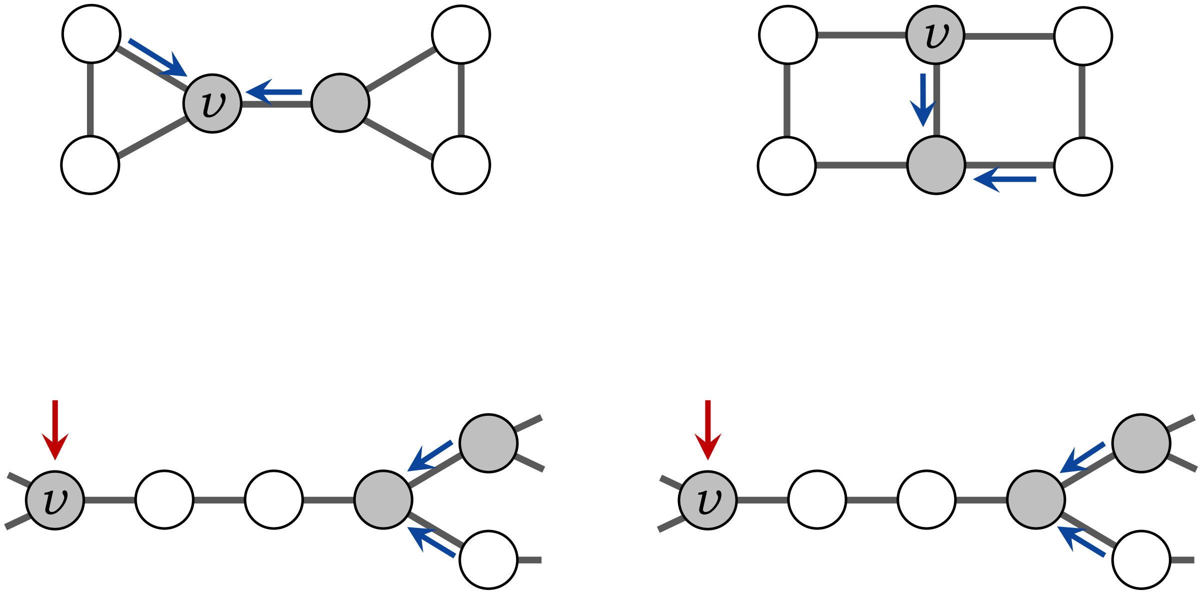
Limits of GNNs



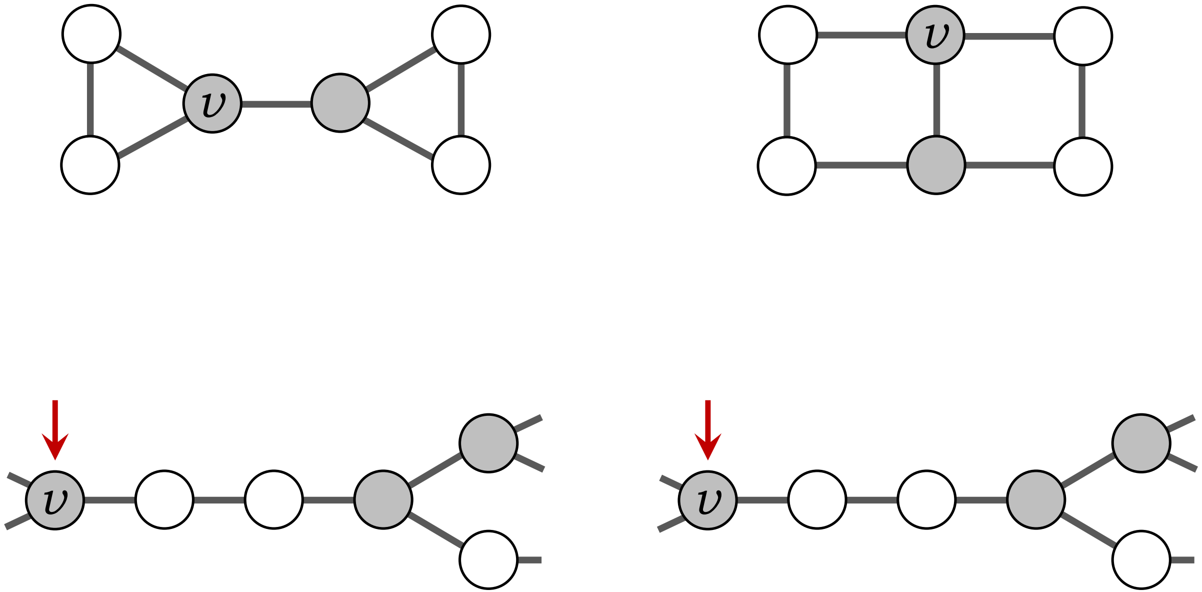
Limits of GNNs



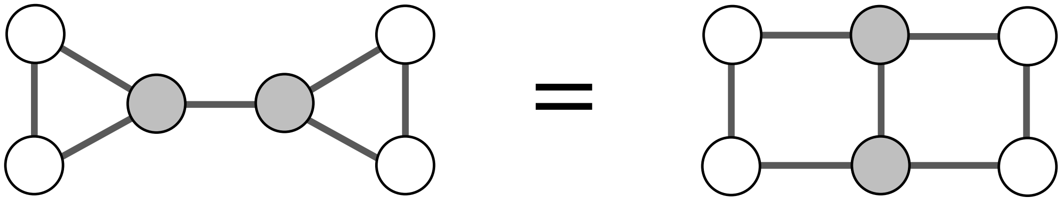
Limits of GNNs



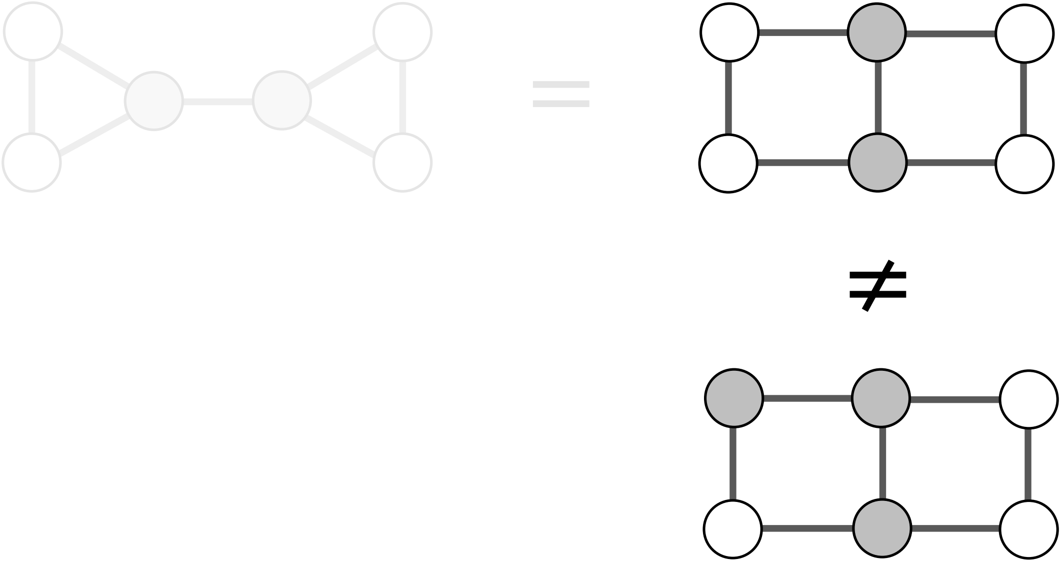
Limits of GNNs



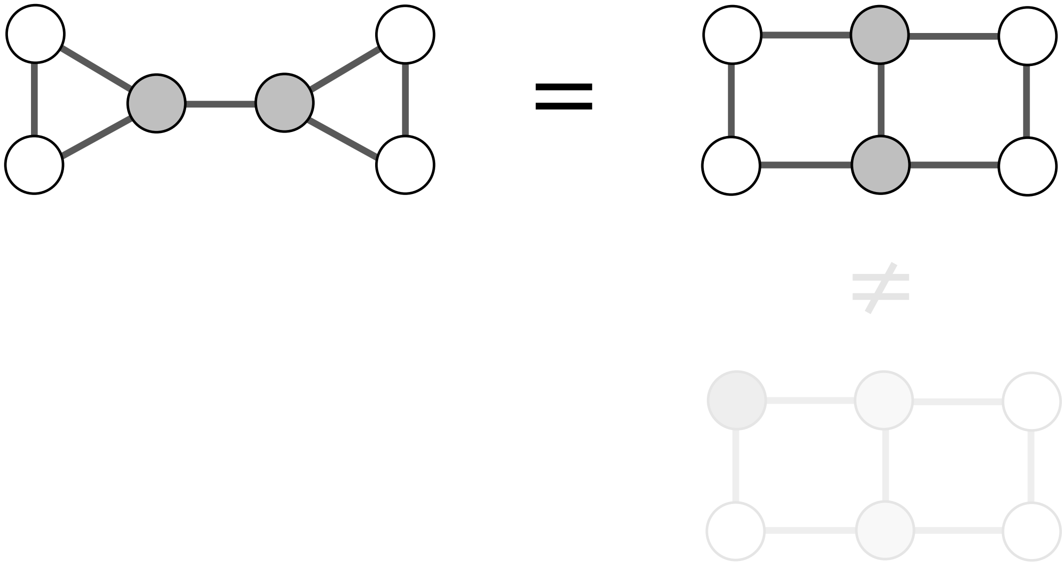
Graph Neural Networks



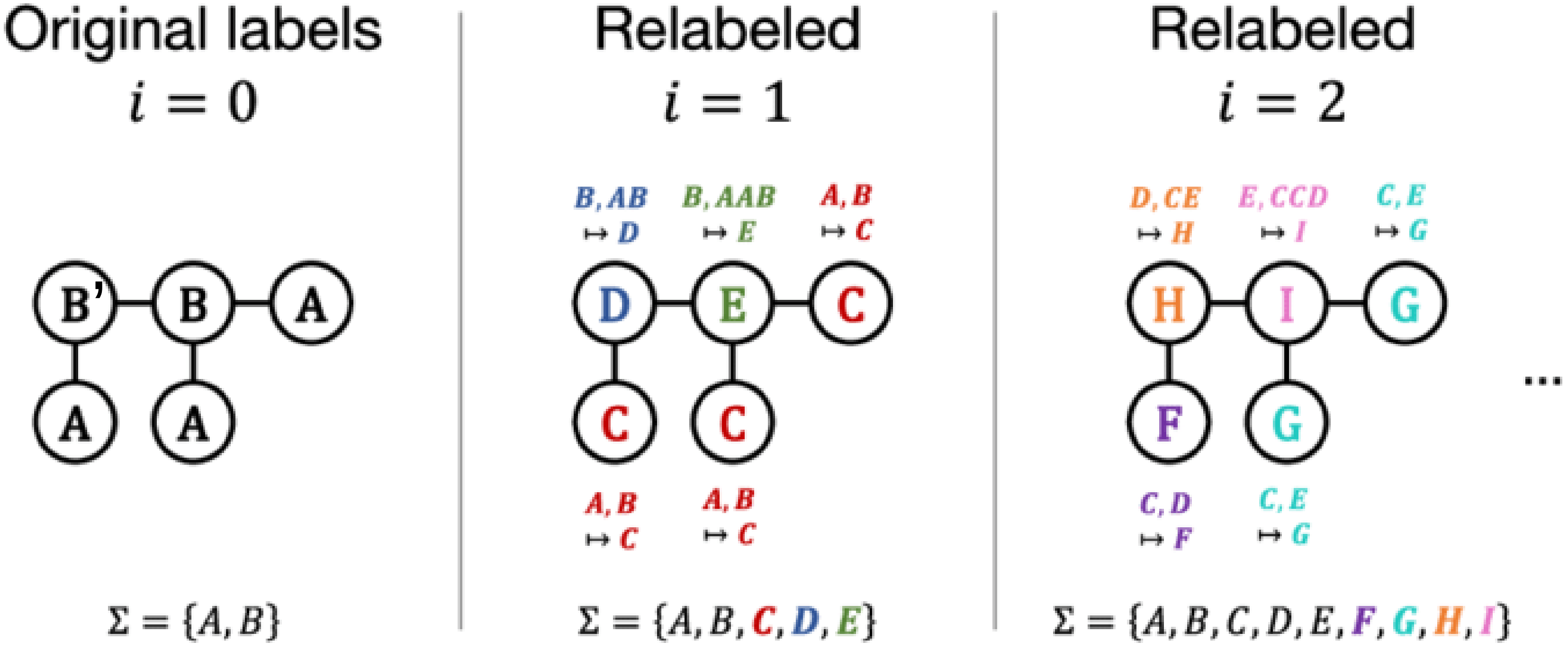
Graph Neural Networks



Graph Neural Networks



Weisfeiler-Lehman Graph Isomorphism Test



More Expressive GNNs?

→ run GNN on metagraph

→ extend GNN model

→ add random features

→ ***DropGNN:*** *GNNs with dropouts*

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

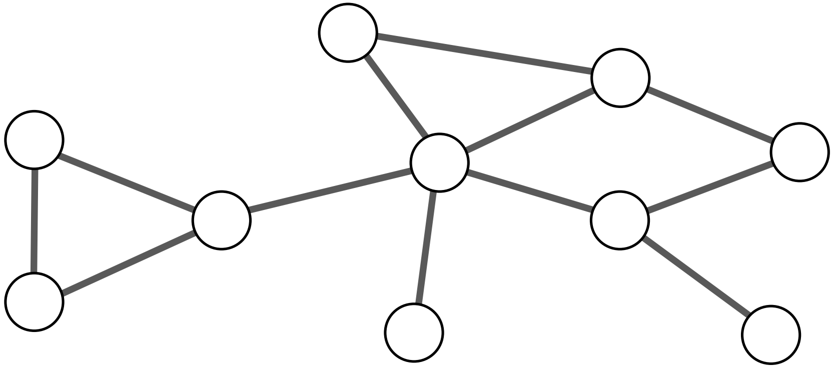
Abstract

This paper studies Dropout Graph Neural Networks (DropGNNs), a new approach that aims to overcome the limitations of standard GNN frameworks. In DropGNNs, we execute multiple runs of a GNN on the input graph, with some of the nodes randomly and independently dropped in each of these runs. Then, we combine the results of these runs to obtain the final result. We prove that DropGNNs can distinguish various graph neighborhoods that cannot be separated by message passing GNNs. We derive theoretical bounds for the number of runs required to ensure a reliable distribution of dropouts, and we prove several properties regarding the expressive capabilities and limits of DropGNNs. We experimentally validate our theoretical findings on expressiveness. Furthermore, we show that DropGNNs perform competitively on established GNN benchmarks.

GNNs with Dropouts

Multiple runs of the GNN

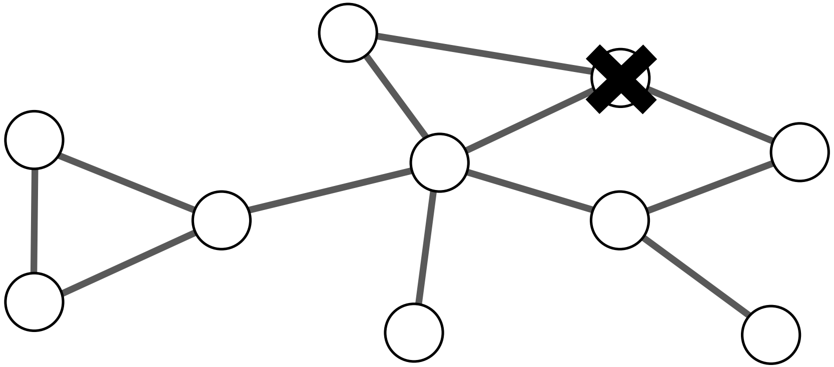
Each node removed with probability *p* independently



GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

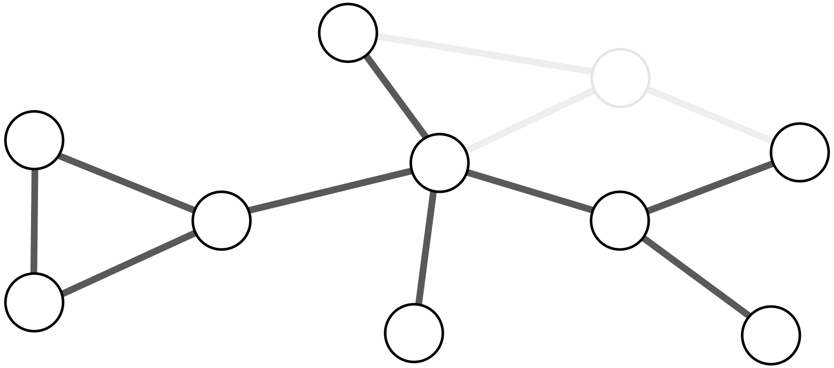


Run #1

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

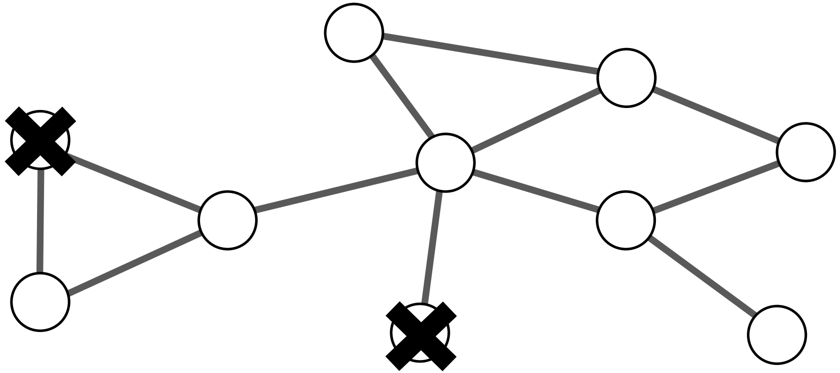


Run #1

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

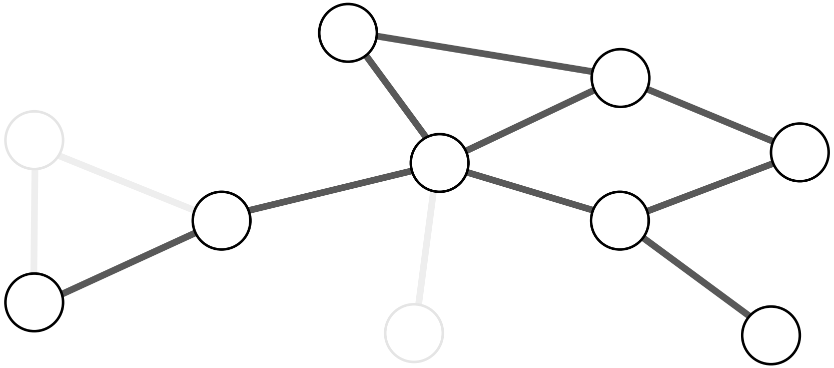


Run #2

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

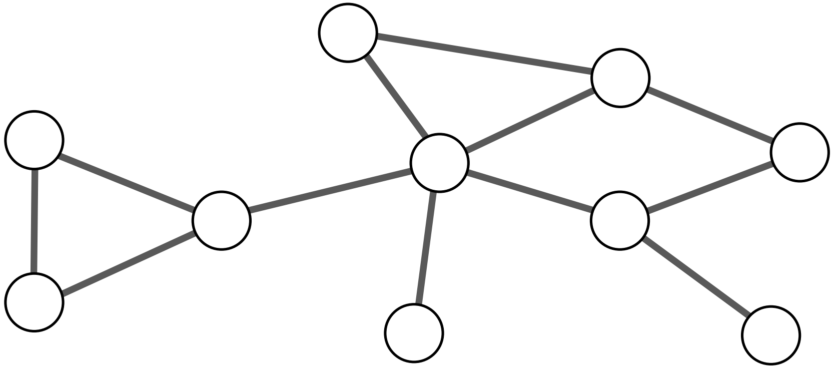


Run #2

GNNs with Dropouts

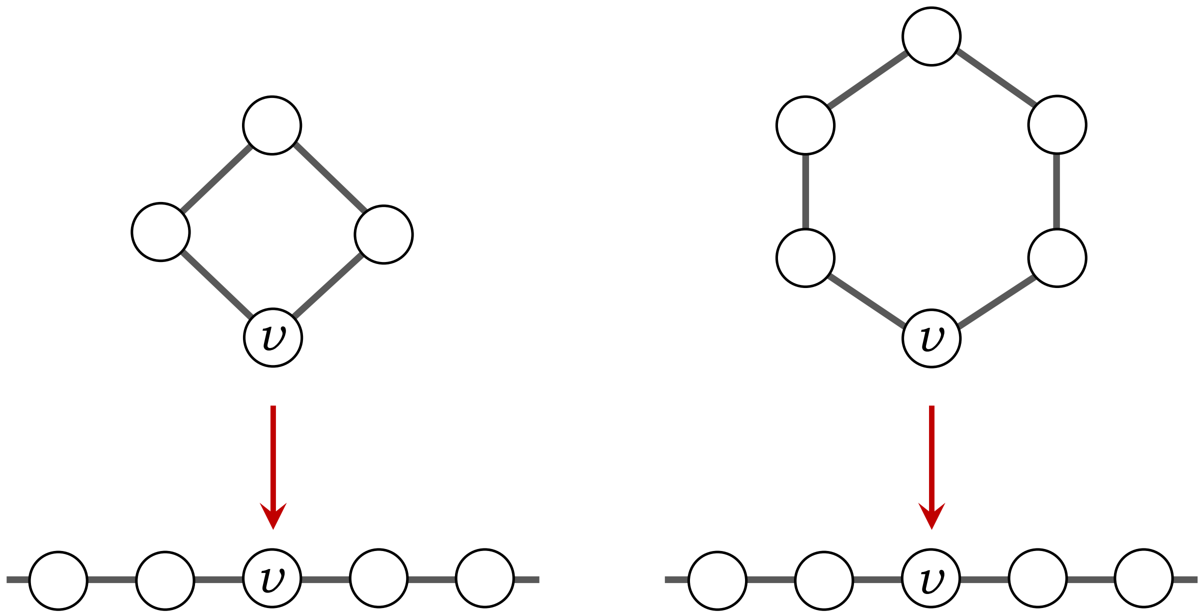
Multiple runs of the GNN

Each node removed with probability *p* independently

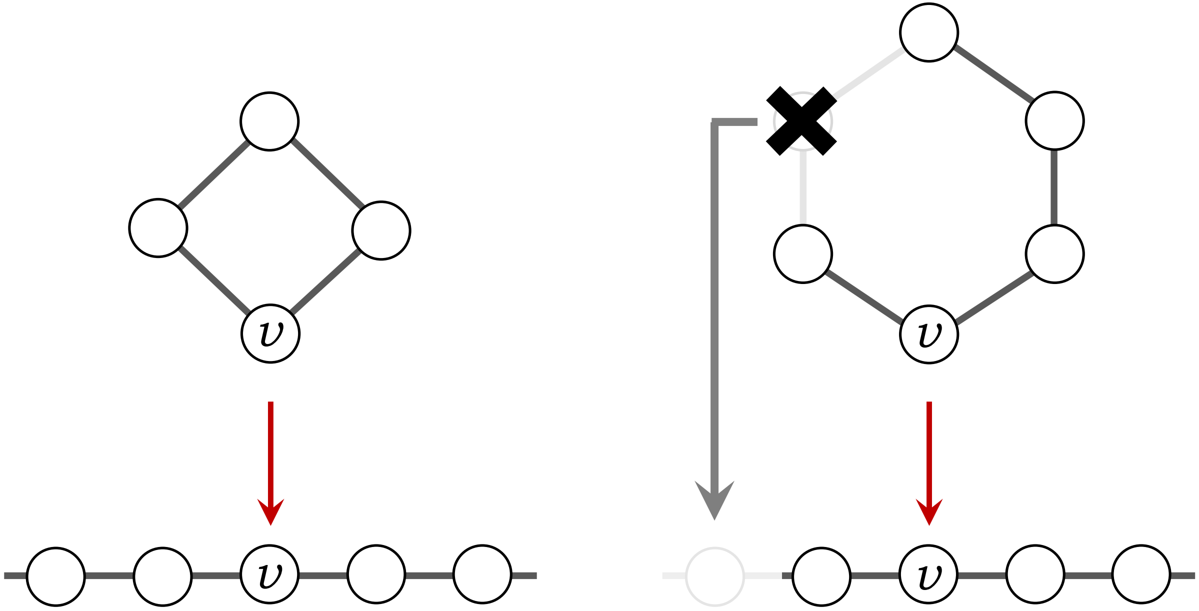


Run #3

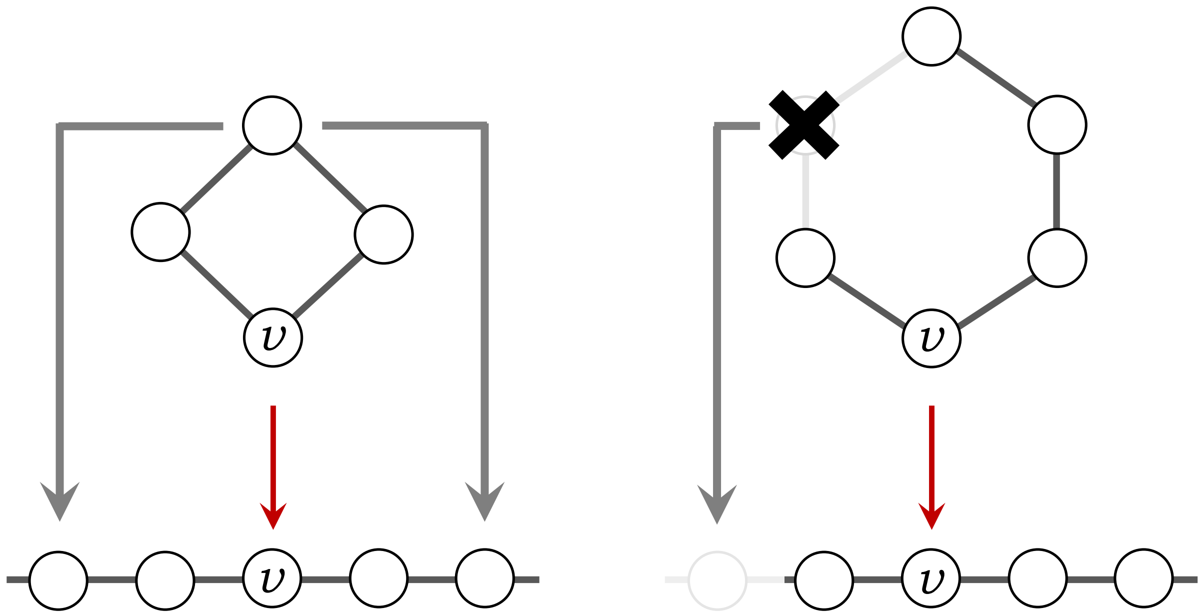
GNNs with Dropouts



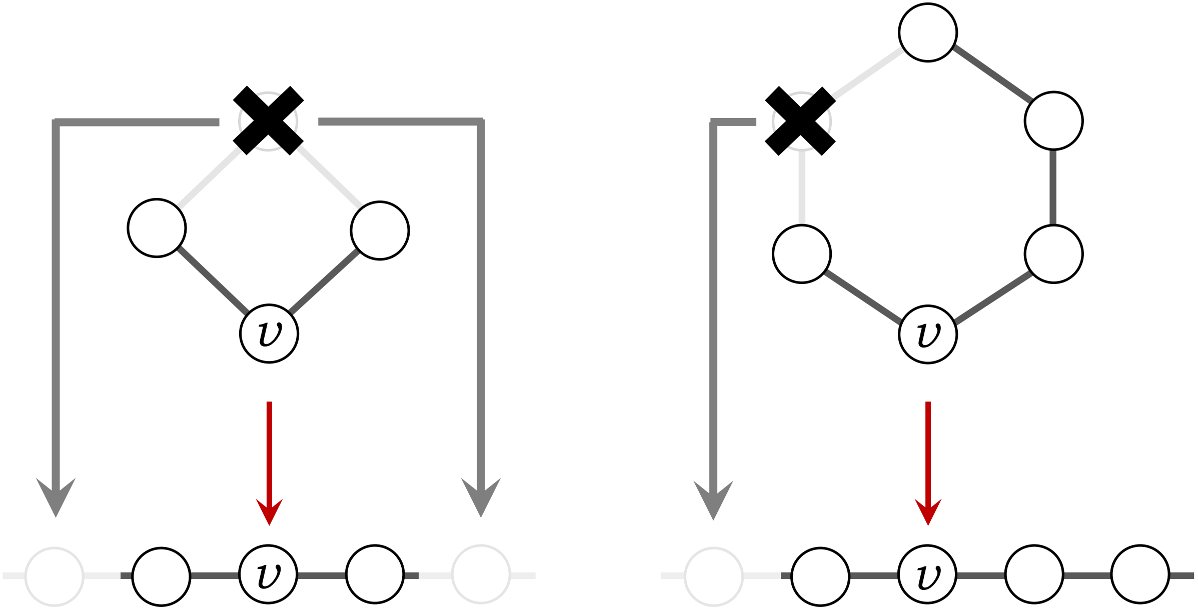
GNNs with Dropouts



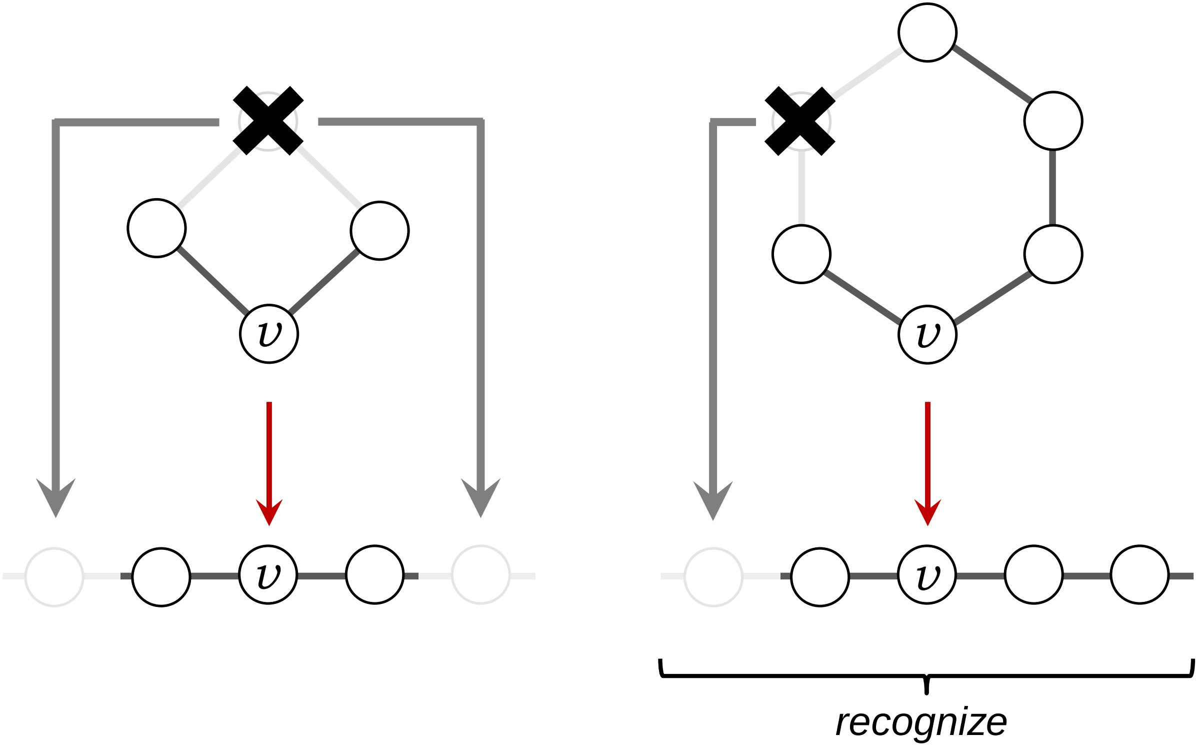
GNNs with Dropouts



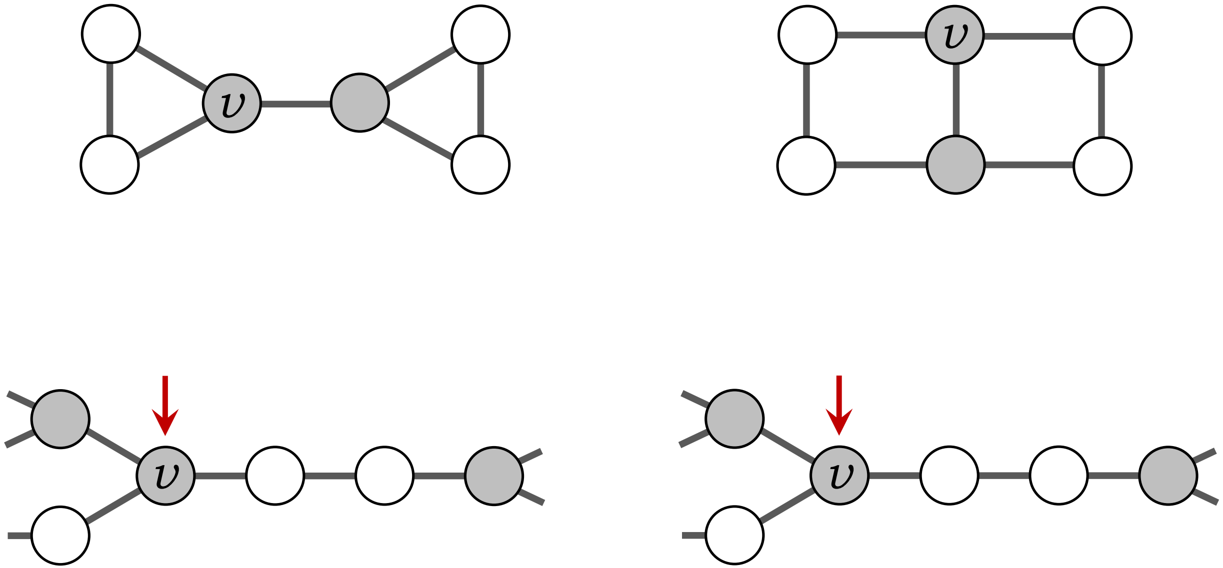
GNNs with Dropouts



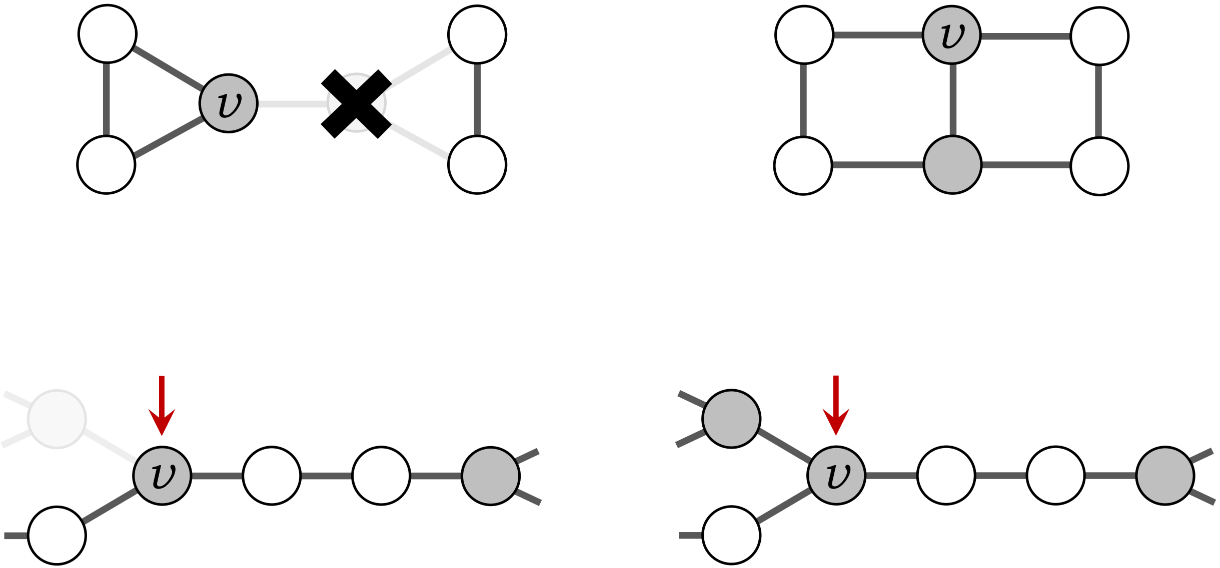
GNNs with Dropouts



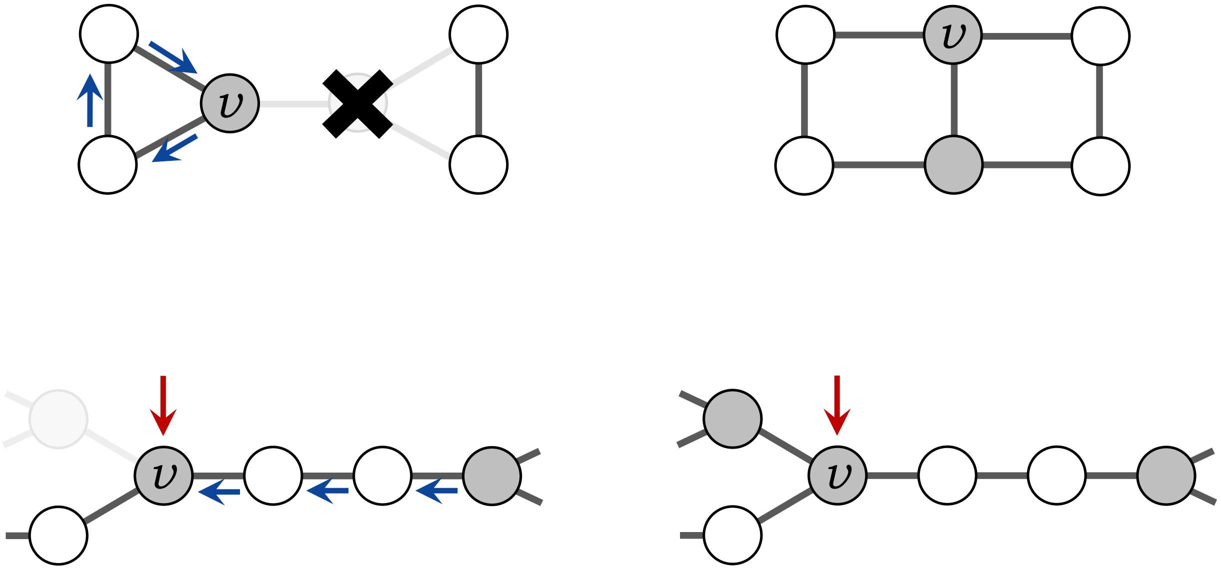
GNNs with Dropouts



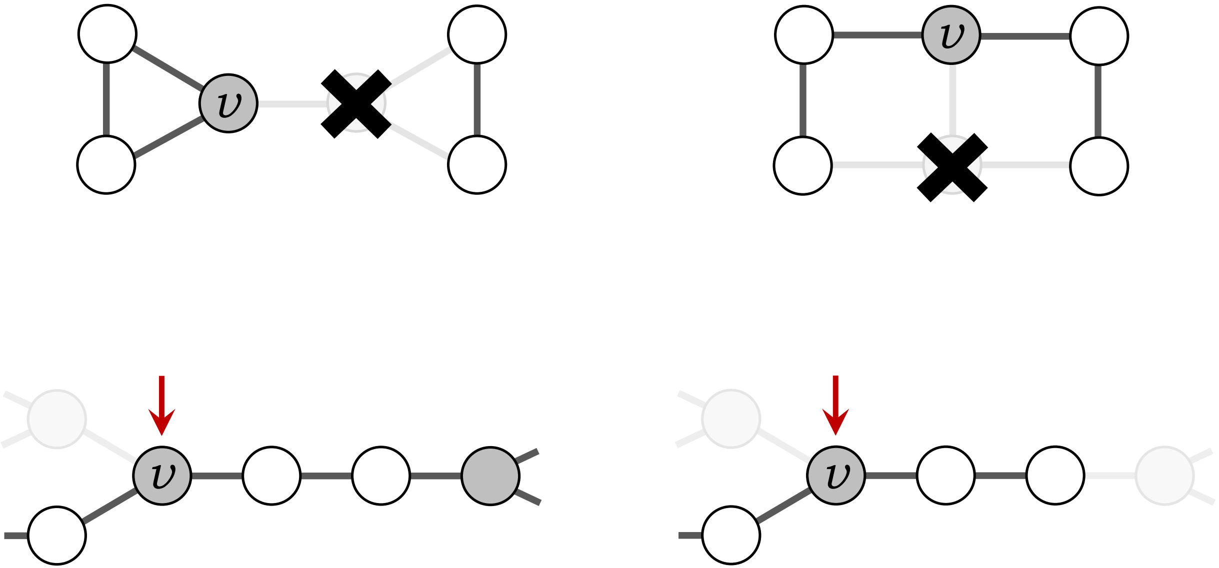
GNNs with Dropouts



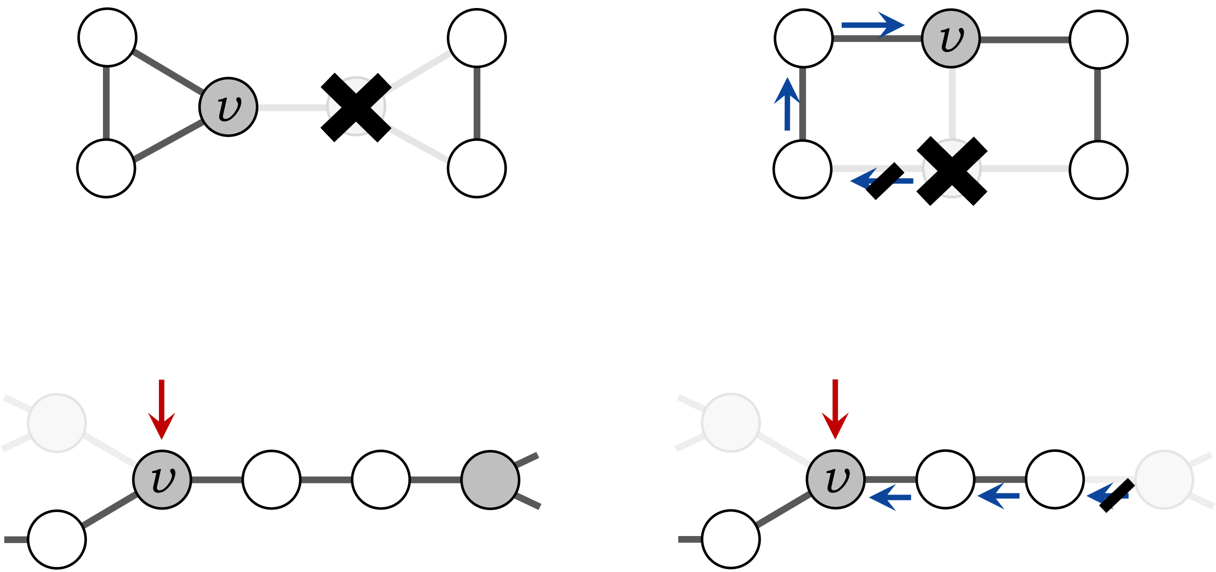
GNNs with Dropouts



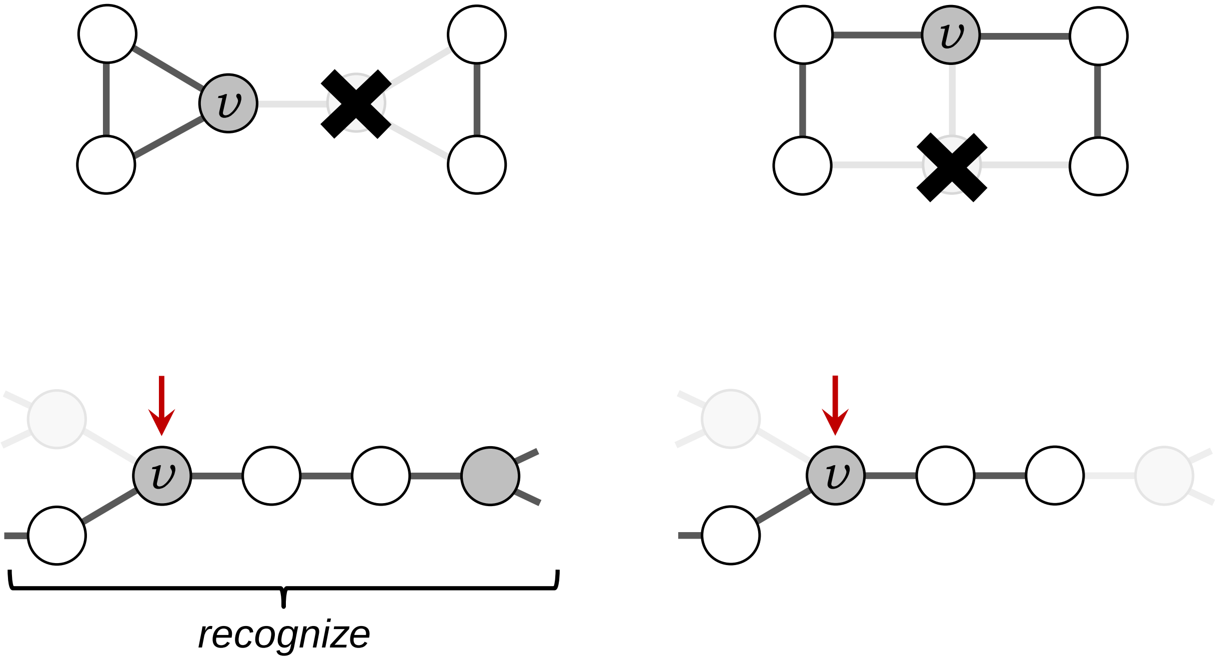
GNNs with Dropouts



GNNs with Dropouts



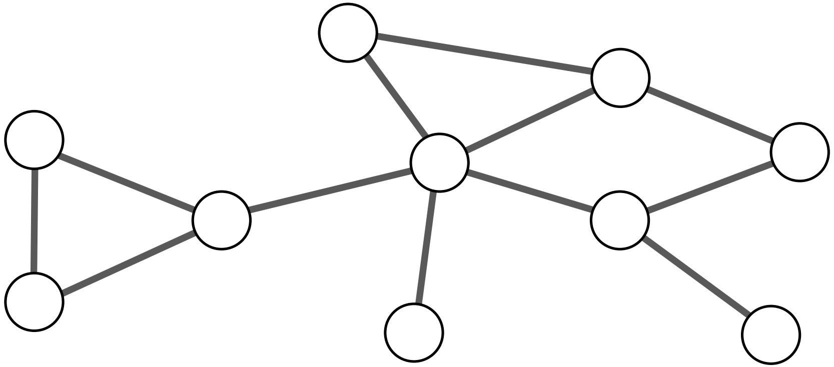
GNNs with Dropouts



GNNs with Dropouts

Multiple runs of the GNN

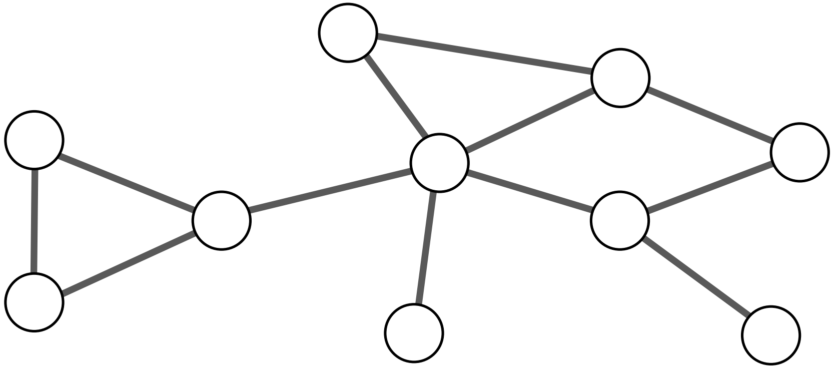
Each node removed with probability *p* independently



GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

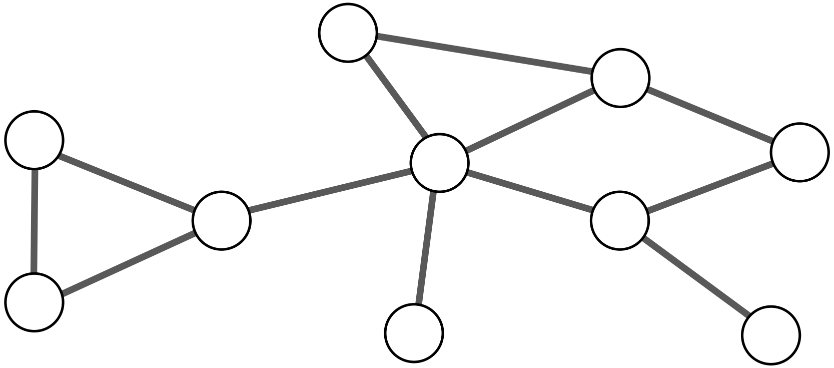


$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

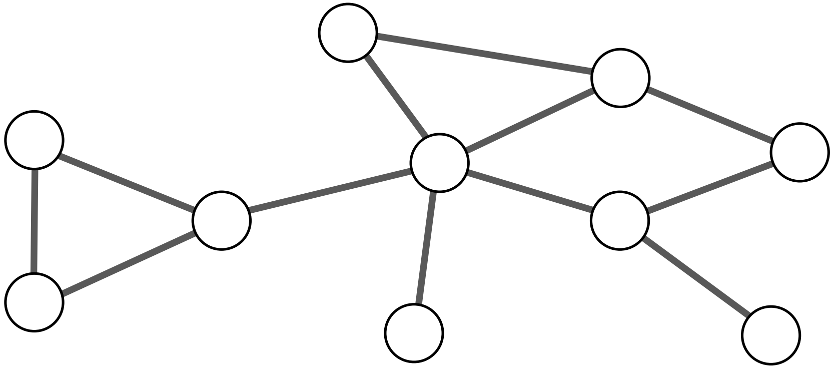


$$h_v = \textsc{RunAggregate}\left(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]}\right)$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

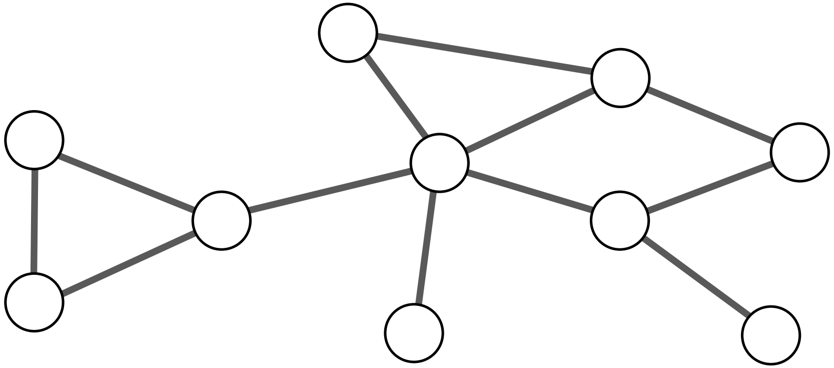


$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently



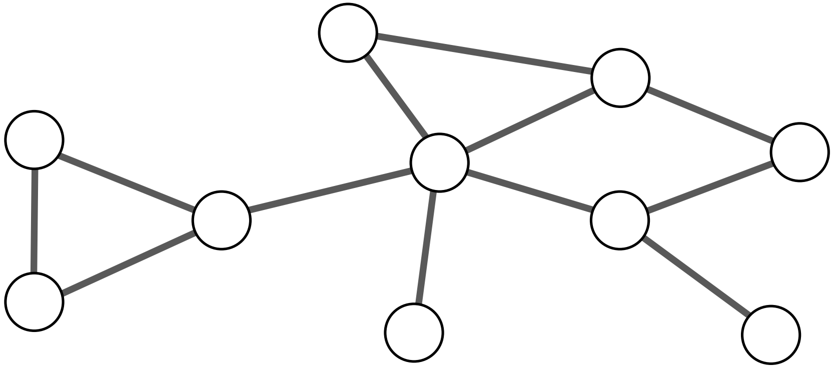
$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots, h_v^{[r]})$$

GNNs with Dropouts

Multiple runs of the GNN

Each node removed with probability *p* independently

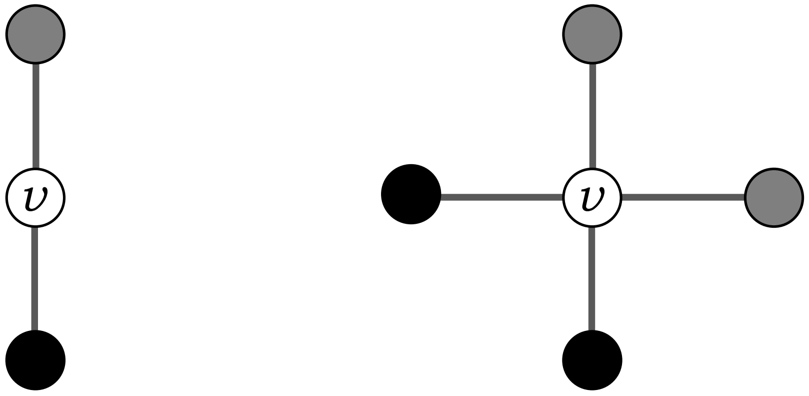
both training and testing!



$$h_v = \textsc{RunAggregate}\,(h_v^{[1]}, h_v^{[2]}, \ldots , h_v^{[r]})$$

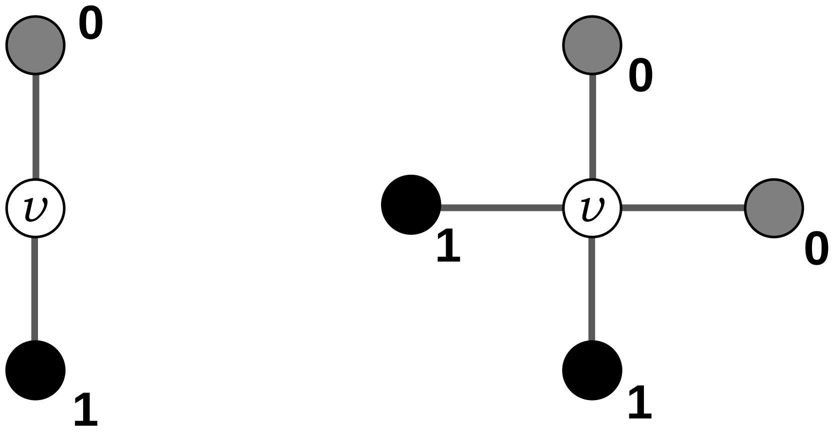
GNNs with Dropouts

MEAN aggregation of neighbors



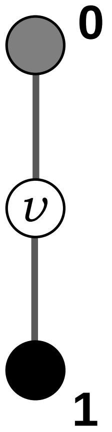
GNNs with Dropouts

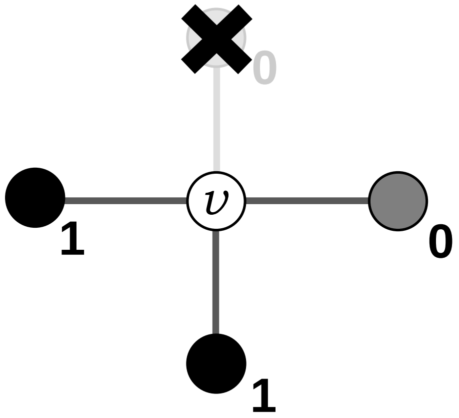
MEAN aggregation of neighbors



GNNs with Dropouts

MEAN aggregation of neighbors

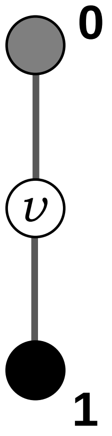




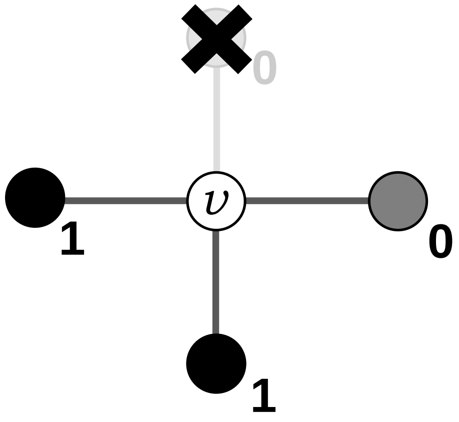
MEAN = 0.66

GNNs with Dropouts

MEAN aggregation of neighbors



MEAN ∈ {0, 0.5, 1}



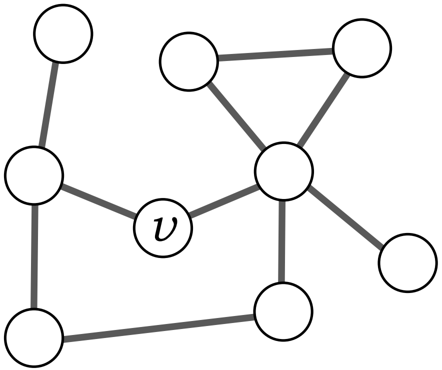
MEAN = 0.66

DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

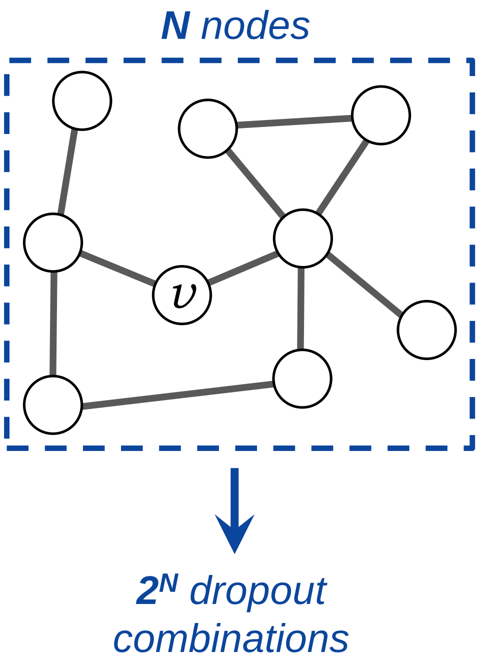


DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead



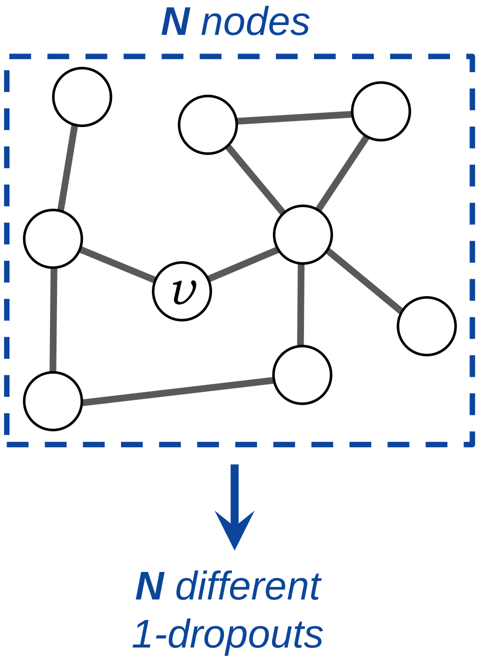
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



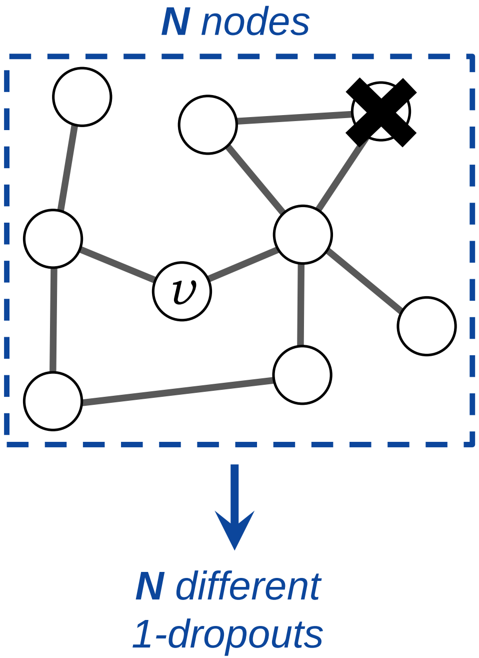
DropGNN with 1-dropouts

More runs:

\+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



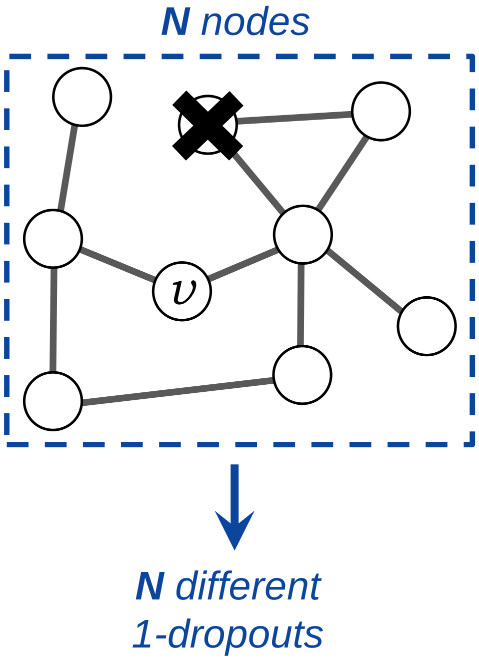
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



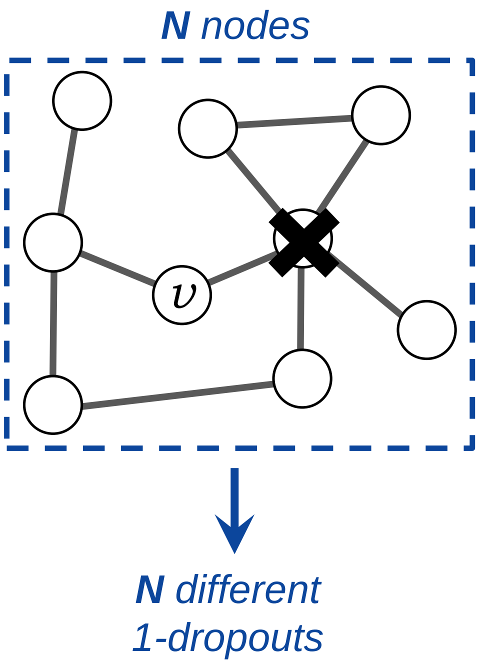
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



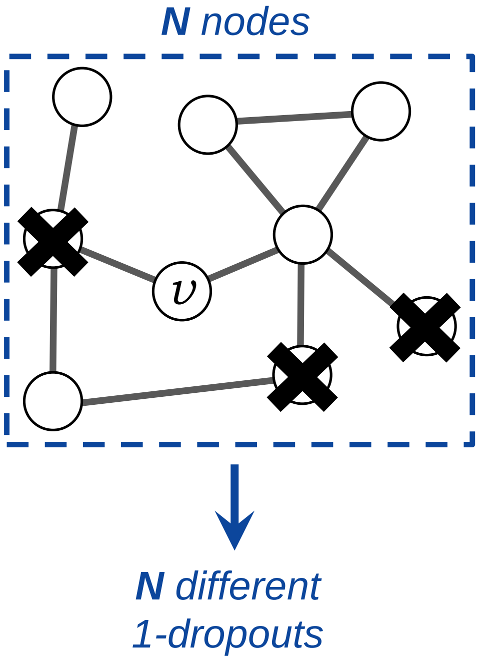
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



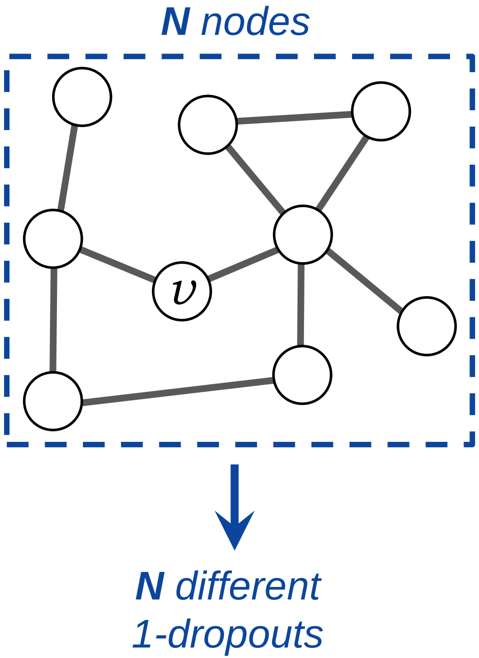
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

Observe every *1-dropout*



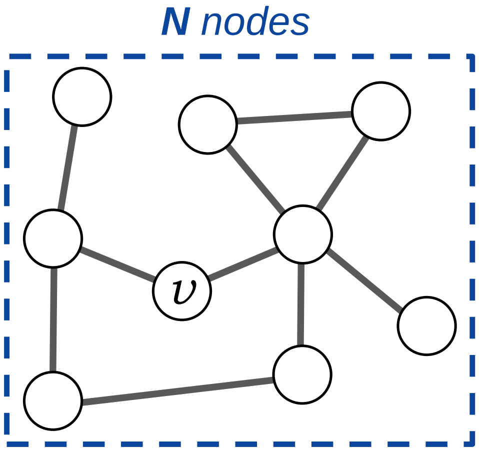
DropGNN with 1-dropouts

More runs:

+ more stable distribution

– more runtime overhead

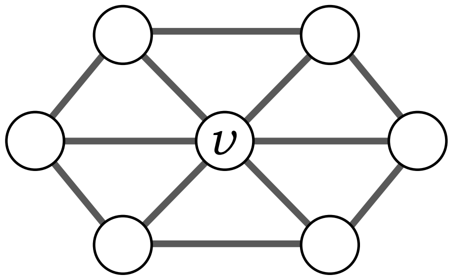
Observe every *1-dropout*



Theorem: if *#runs* ≈ *N* · log *N*, then we observe every 1-dropout with high probability.

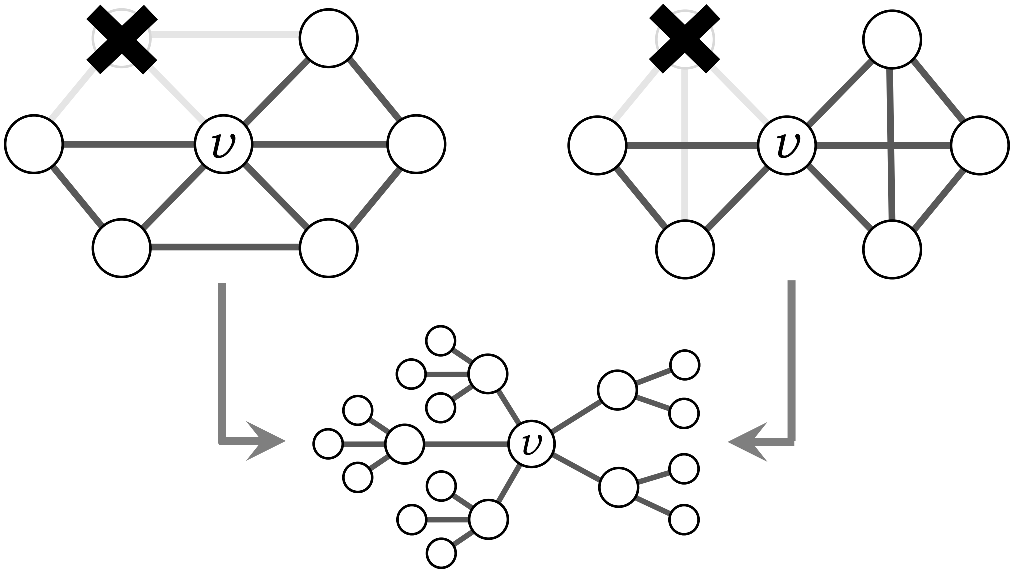
DropGNN with 1-dropouts

Theorem: There are graphs that cannot be distinguished from 1-dropouts only.



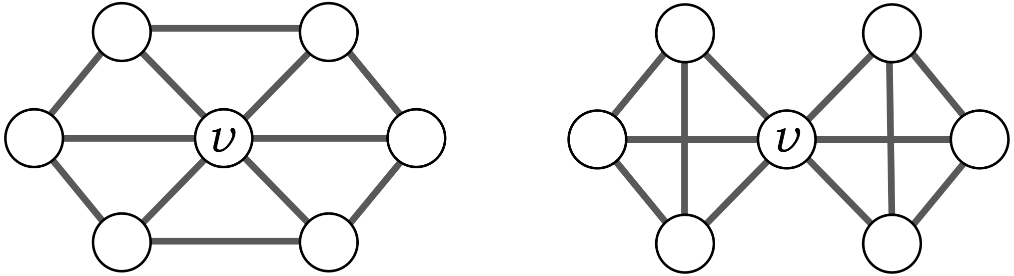
DropGNN with 1-dropouts

Theorem: There are graphs that cannot be distinguished from 1-dropouts only.



DropGNN with 1-dropouts

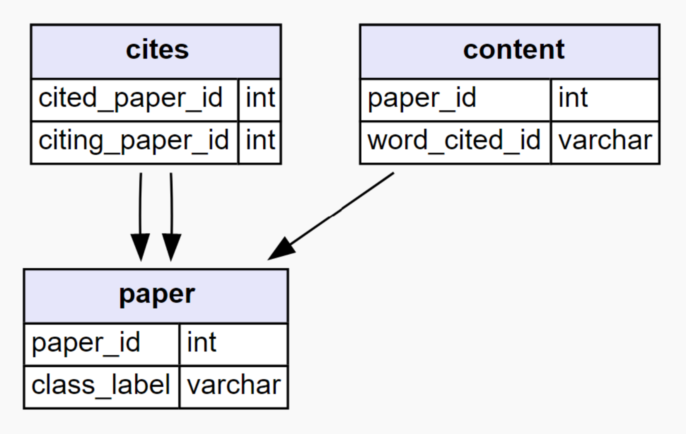
Theorem: There are graphs that cannot be distinguished from 1-dropouts only.

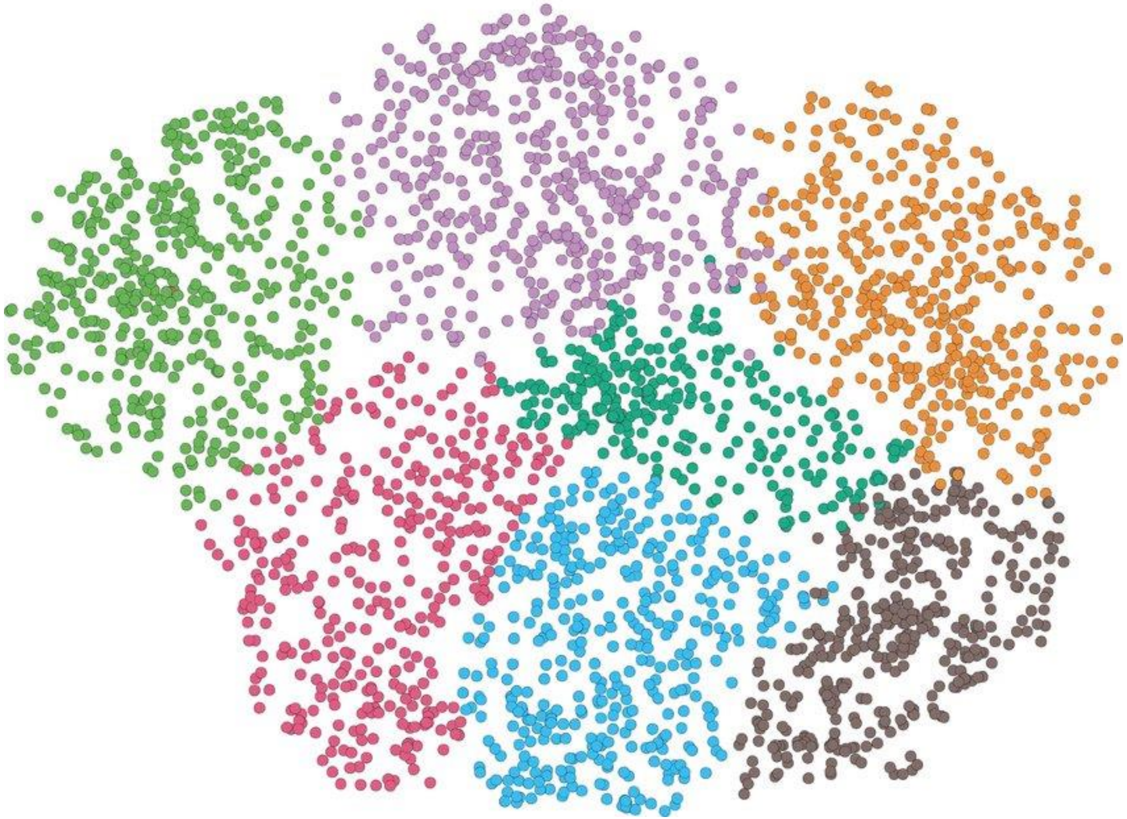


Theorem: in DropGNNs with *port numbers,* any two graphs can be distinguished from 1-dropouts.

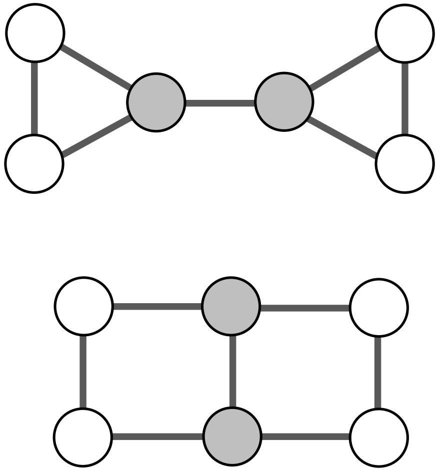

BAD DATA

Example: CORA Benchmark





Example: CORA Benchmark



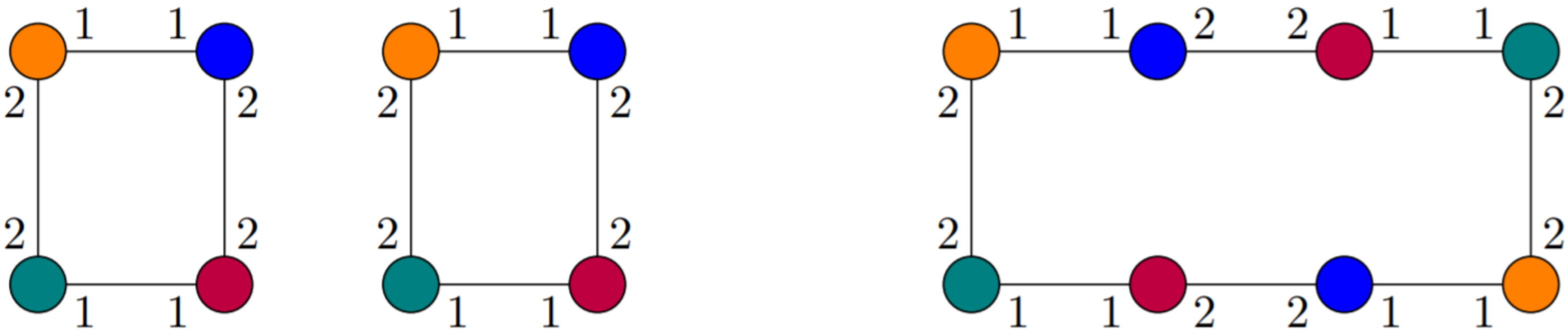
Title	Keywords	Neighbor Labels	Neighbor Keywords
Primes is in P	…	Crypto, …	…

Experiments: QM9 dataset

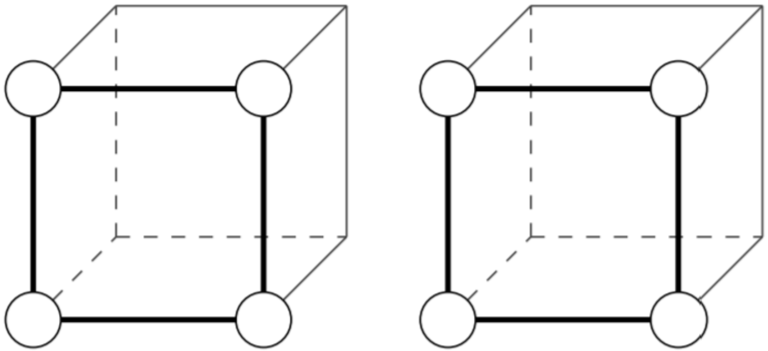
Property	Unit	GNN	DropGNN	PPGNN
μ	Debye	0.358	**0.077**	0.0934
α	$Bohr^3$	0.89	**0.238**	0.318
ϵ_{HOMO}	Hartree	0.00541	0.00235	**0.00174**
ϵ_{LUMO}	Hartree	0.00623	0.00241	**0.0021**
$\boldsymbol{\Delta\epsilon}$	Hartree	0.0066	0.0044	**0.0029**
$\langle R^2 \rangle$	$Bohr^2$	28.5	**0.472**	3.78
ZPVE	Hartree	0.00216	**0.000153**	0.000399
U_0	Hartree	2.05	0.251	**0.022**
U	Hartree	2.0	0.146	**0.0504**
H	Hartree	2.02	0.0845	**0.0294**
G	Hartree	2.02	**0.188**	0.24
C_v	cal/(mol K)	0.42	0.0740	**0.0144**

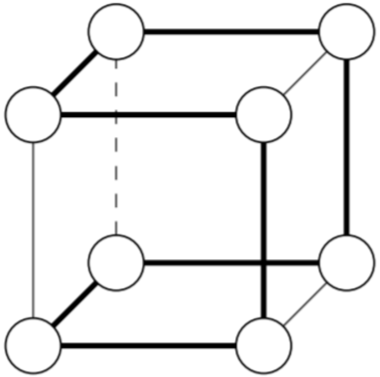
Other Extension Ideas?

Port Numbers

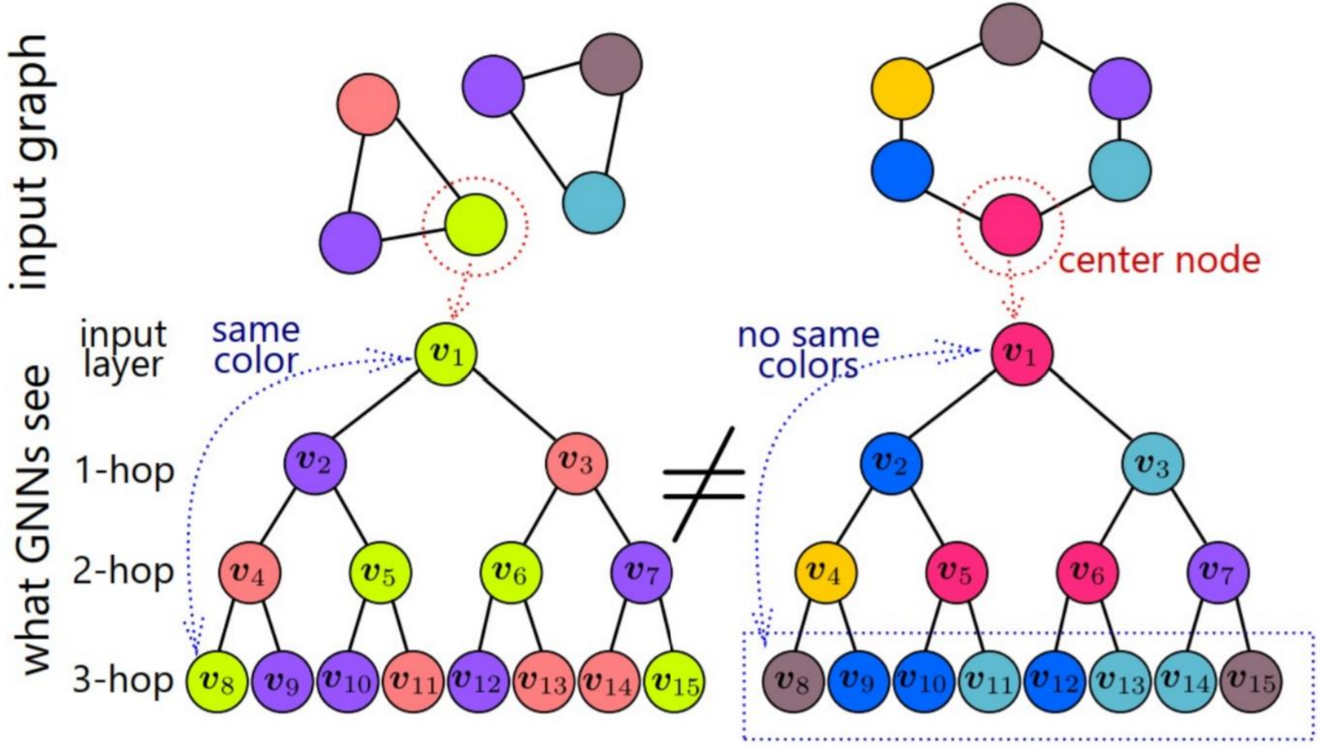


Angle Features

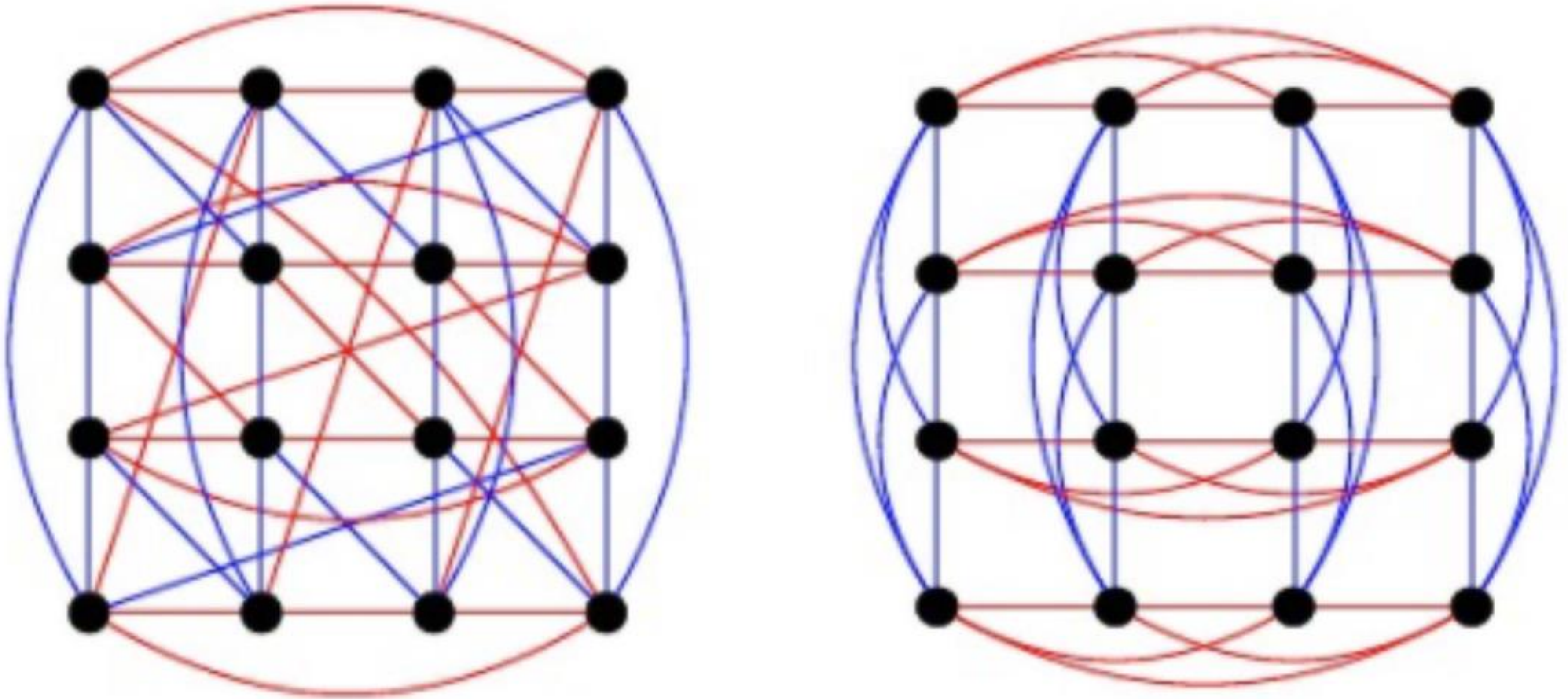




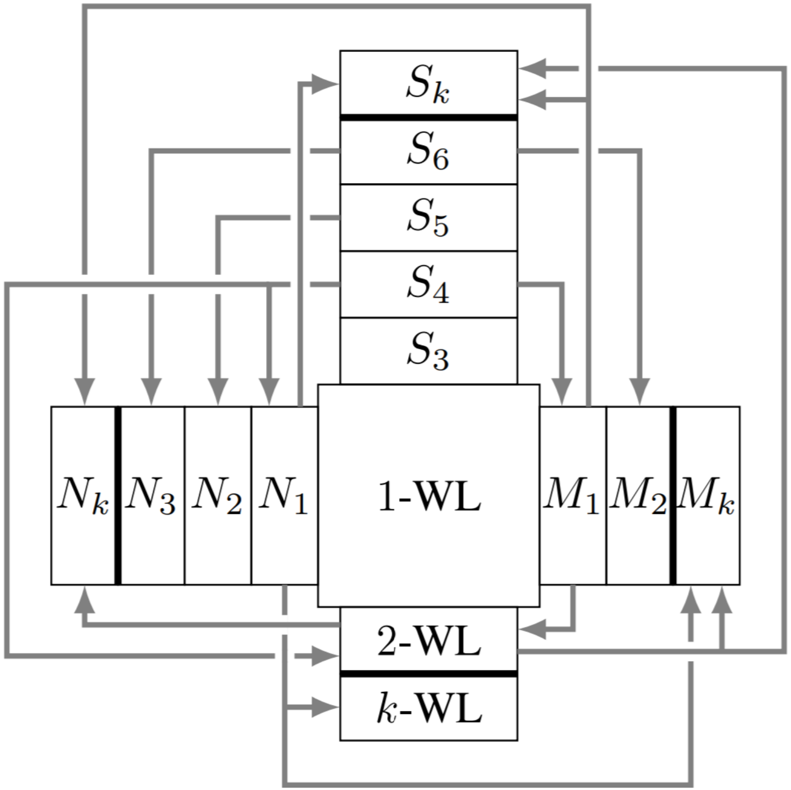
Random Features



2-WL

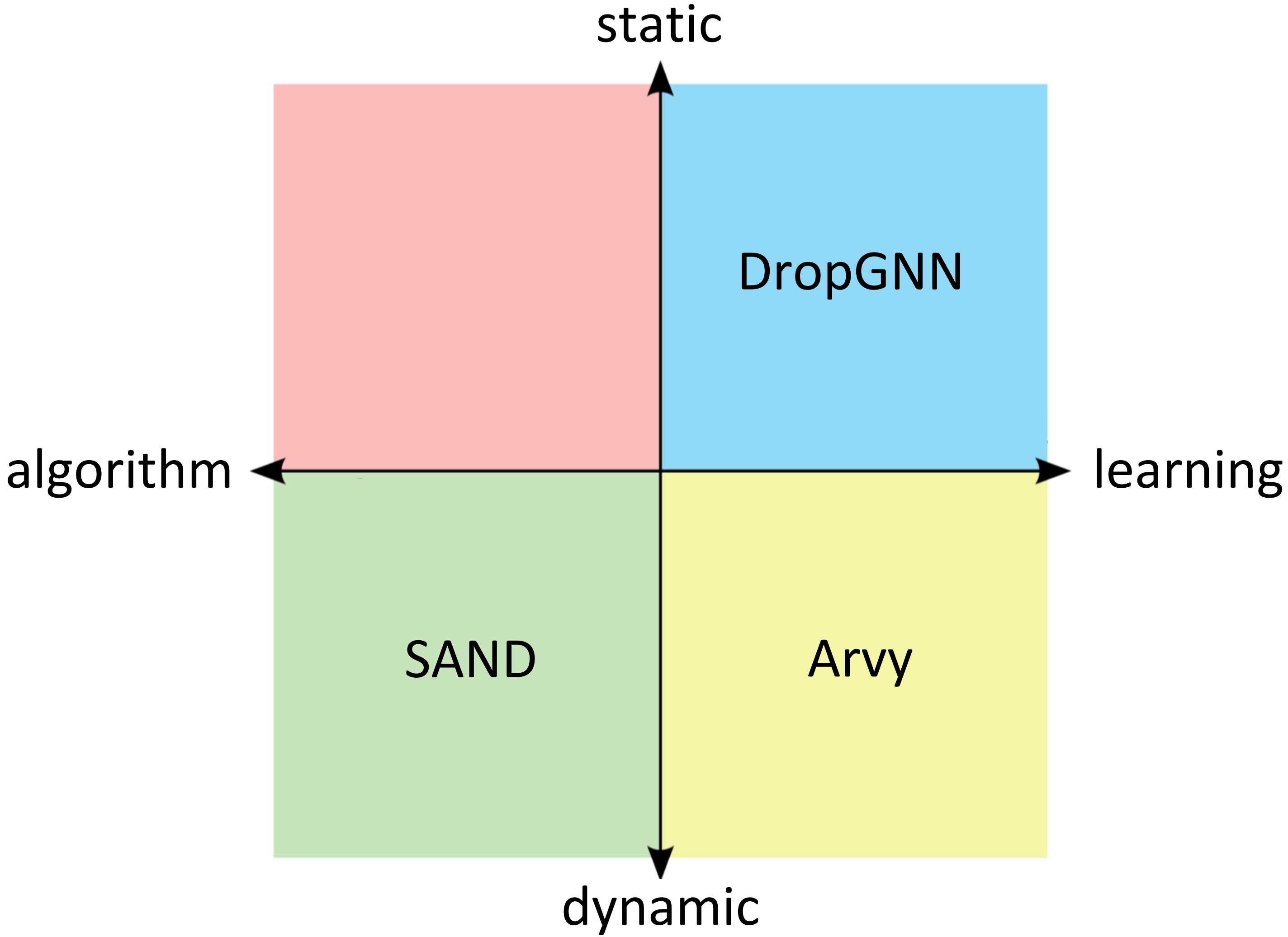


Comparisons of Extensions



Open Questions

- **Theory:** characterization of graphs that can be distinguished by extensions?
- **Experiments:** other applications where the graph structure is crucial?
- **General:** similar approach in other deep learning areas?


static
DropGNN
algorithm
learning
SAND
Arvy
dynamic

Thank You!

Questions & Comments?

Roger Wattenhofer, ETH Zurich, www.disco.ethz.ch