Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning

Fabien Geyer, Georg Carle

Monday 20th August, 2018

ACM SIGCOMM Workshop Big-DAMA'18, Budapest, Hungary

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Motivation

Distributed protocols

Today's distributed network protocols

- Manually developed, engineered and optimized
- Sometimes hard to configure to achieve good performance
- Not always adapted to evolving networks and requirements (eg. mobile networks, sensor networks, . . .)

Main research questions

- **Can we automate distributed network protocol design using high-level goals and data?**
- **If yes, can properties such as resilience to faults be included (eg. packet loss)?**

Contribution

- Method for generating protocols using Graph Neural Networks
- Today's focus: routing protocols

Motivation

Why now?

Two recent trends in networking for enabling such data-driven protocols

- More advanced in-network processing resources and capabilities (eg. SDN, P4, DPDK, . . .) + flexibility
- Data-driven networks and data-driven protocols → **See this year's SIGCOMM workshops**

Motivation

Why now?

Two recent trends in networking for enabling such data-driven protocols

- More advanced in-network processing resources and capabilities (eg. SDN, P4, DPDK, . . .) + flexibility
- Data-driven networks and data-driven protocols → **See this year's SIGCOMM workshops**

A more general problem in Artificial Intelligence

- **Research question: autonomous agents communicating and collaborating to reach a common goal**
- Human-level performance in multiplayer games:
  - DeepMind: 2vs2 Quake 3 Capture The Flag *(July 2018)*
    → `https://deepmind.com/blog/capture-the-flag/`
  - OpenAI: 5vs5 Dota 2 *(August 2018)*
    → `https://blog.openai.com/openai-five/`

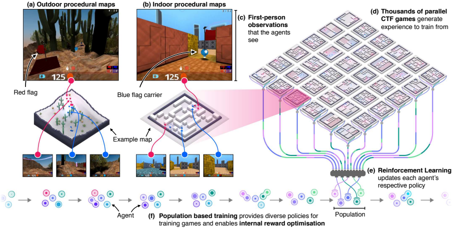


Figure 1: Overview of DeepMind's Quake 3 challenge (source: `https://arxiv.org/abs/1807.01281`)

Outline

Introduction

Definition

Distributed network protocols

- Distributed nodes need to solve a common high-level goal
- Nodes need to share some information to achieve the goal
- Examples: routing, congestion control, load balancing, content distribution, . . .

Target protocol behavior for this talk: **simplified version of OSPF** (Open Shortest Path First)

Introduction

Main assumptions

Protocol properties and requirements

- Routing follows a predetermined path-finding scheme (e.g. shortest path)
- Protocol needs to support routers entering and leaving the network
- Protocol needs to be resilient to packet loss
- Should work on any topology

Assumptions

- Routers start with no information about the network topology
- Routers have only their own local view of the network and need to exchange information

Introduction

General idea

- Represent the network as a graph
  - Nodes $\leftrightarrow$ Routers *(+ some extra nodes)*
  - Edges $\leftrightarrow$ Physical links
  - Data exchange between nodes $\leftrightarrow$ Communication between routers
- Use a neural network architecture able to process graphs
- Train on dataset emulating the network protocol's goal

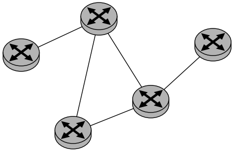


Figure 2: Computer network

$\leftrightarrow$

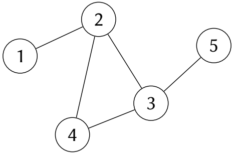


Figure 3: Graph representation

$\leftrightarrow$

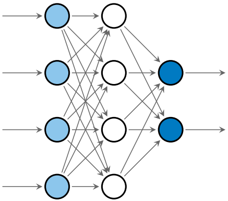


Figure 4: Neural network

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

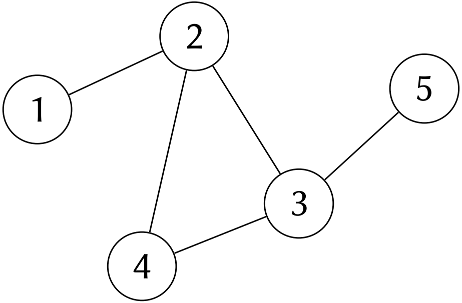


Figure 5: Example graph

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$

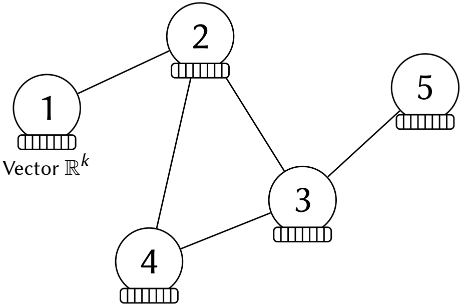


Figure 5: Hidden representations

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors

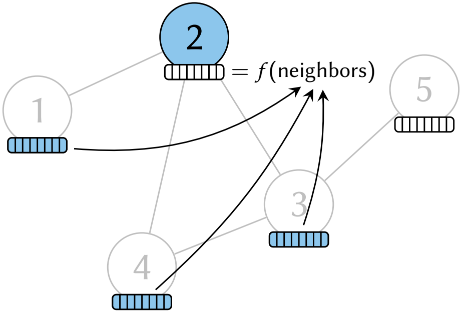


Figure 5: Relationship between hidden representations

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are fixed points: $\mathbf{h}_v = f\left(\{\mathbf{h}_u \mid u \in Nbr(v)\}\right)$

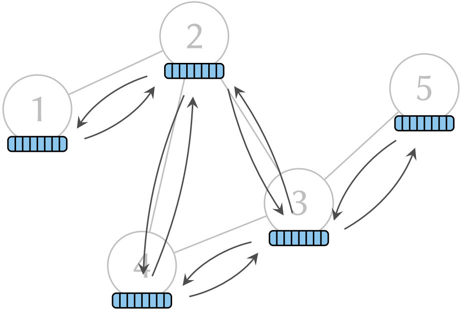


Figure 5: Relationship between hidden representations

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are fixed points: $\mathbf{h}_v = f\left(\{\mathbf{h}_u \mid u \in Nbr(v)\}\right)$

Implementation

- The vectors are initialized with the nodes' input features

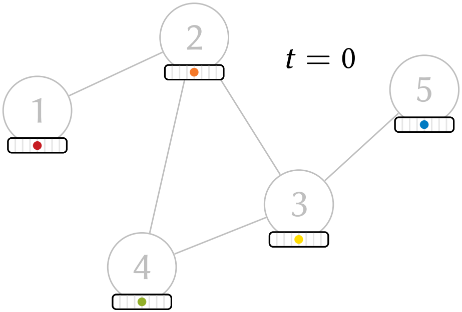


Figure 5: Hidden representations initialization

Graph Neural Networks

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$
- ... computed according to the vector of its neighbors
- ... and are fixed points: $\mathbf{h}_v = f\left(\{\mathbf{h}_u \mid u \in Nbr(v)\}\right)$

Implementation

- The vectors are initialized with the nodes' input features
- They are iteratively propagated between neighbors

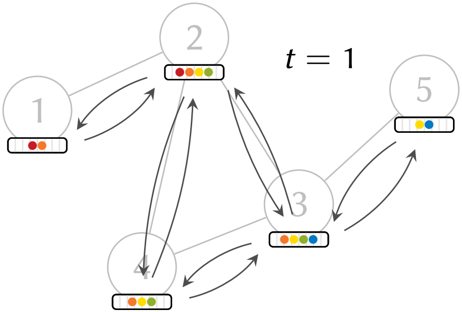


Figure 5: Hidden representations propagation

Graph Neural Networks

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$
- . . . computed according to the vector of its neighbors
- . . . and are fixed points: $\mathbf{h}_v = f\left(\{\mathbf{h}_u \mid u \in Nbr(v)\}\right)$

Implementation

- The vectors are initialized with the nodes' input features
- They are iteratively propagated between neighbors
- . . . until a fixed point is found or for a fixed number of iterations

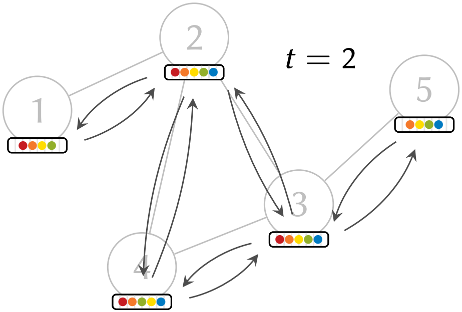


Figure 5: Hidden representations propagation

Graph Neural Networks

Main concept

Graph Neural Networks [Scarselli et al., 2009] and related neural network architectures are able to process general graphs and predict features of nodes $\mathbf{o}_v$

Principle

- Each node has a hidden representation vectors $\mathbf{h}_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are fixed points: $\mathbf{h}_v = f\left(\{\mathbf{h}_u \mid u \in Nbr(v)\}\right)$

Implementation

- The vectors are initialized with the nodes' input features
- They are iteratively propagated between neighbors
- ...until a fixed point is found or for a fixed number of iterations
- Those vectors are then used for the final prediction: $\mathbf{o}_v = g\left(\mathbf{h}_v\right)$

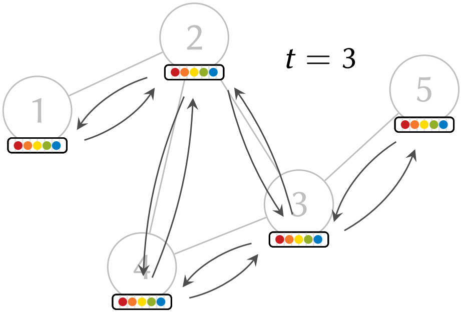


Figure 5: Hidden representations fixed point

Graph Neural Networks

Details

Implementation

- f and g are neural networks which need to be trained
- f and g implemented as standard feed-forward neural networks in [Scarselli et al., 2009]
- f also implemented using a Gated Recurrent Unit in [Li et al., 2016]
- GNN extended with edge attention to learn which edges are important [Veličković et al., 2018]

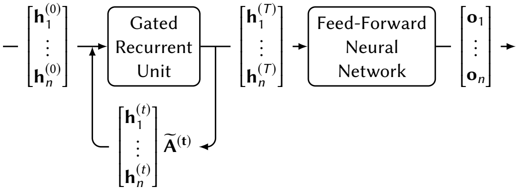


Figure 6: Gated Graph Neural Network architecture

Main advantage of GNNs

- Not restricted to a specific graph (i.e. network topology) type such as size, shape, etc.

Generation of distributed protocols

Basic idea

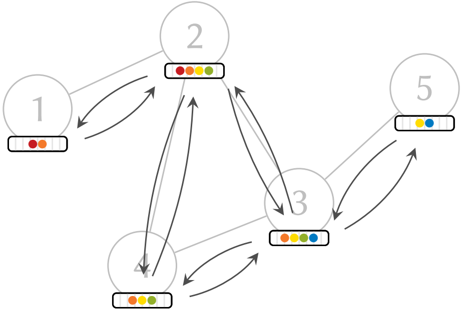


Figure 7: Graph analyzed by the GNN

$\leftrightarrow$

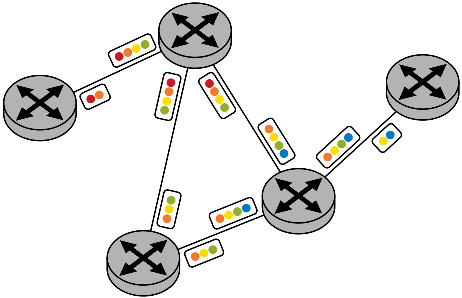


Figure 8: Network topology

1. Nodes in the computer network periodically broadcast their hidden representation vector
2. Periodically process locally the received hidden representations using the *f* function previously trained
3. Go to step 1

Generation of distributed protocols

Goal: Given a destination, routers need to know the next hop, i.e. which output interface to use

Transformation from topology to graph

- Each router is a node with a router identifier as input feature
- Each interface is a node with a binary output feature: given a destination (i.e. router identifier) use interface or not
- Edges correspond to physical links

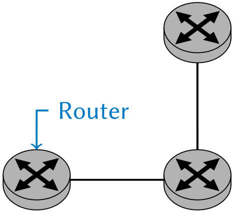


Figure 9: Network topology

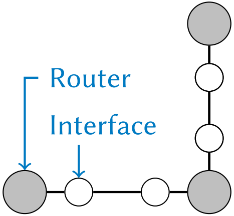


Figure 10: Graph encoding of topology

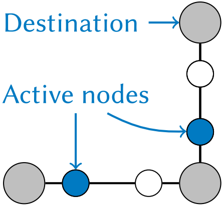


Figure 11: Output features based on queried destination

Generation of distributed protocols

Extension of Graph Neural Networks

- Exchange data about the network topology $\rightarrow \mathbf{h}_n^{(t)}$
- Store a local view of the topology (i.e. the hidden representation vector $\mathbf{h}_n^{(t)}$)
- Query the local view for routing information (i.e. next hop $\mathbf{q}$) $\rightarrow \mathbf{h}_n^{(T)} \odot \mathbf{q} = \mathbf{o}_n$

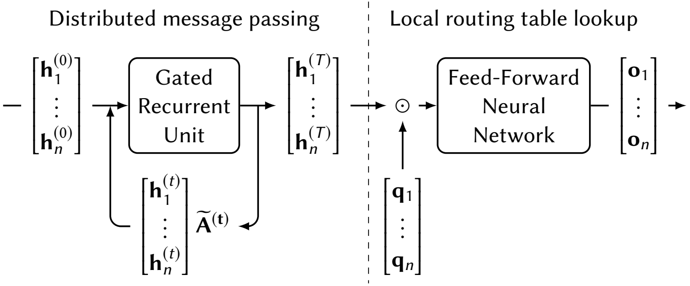


Figure 12: **Graph Query Neural Network**

Numerical evaluation

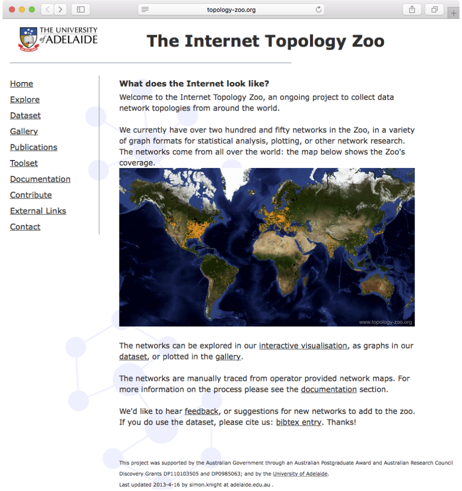
Description

Dataset

- Dataset based on Topology Zoo [Knight et al., 2011][1]
  - Max number of nodes: 20
  - Max hop count: 10
- Randomly add or remove one router in the topologies
- Randomly generate router identifiers
- Total number of generated data points: 40 000

Use-cases

- Shortest-path routing
- Max-min routing

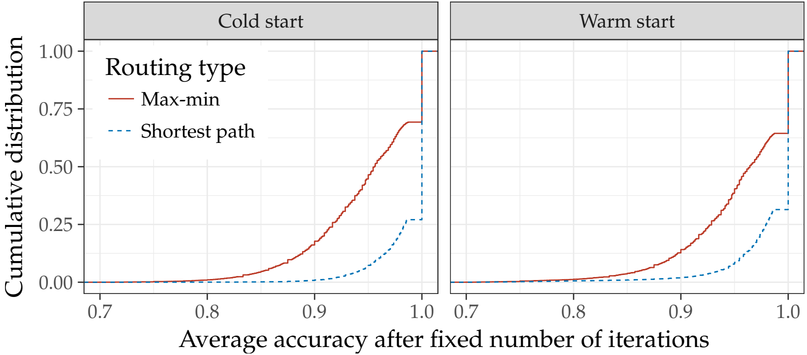


[1] http://www.topology-zoo.org

Numerical evaluation

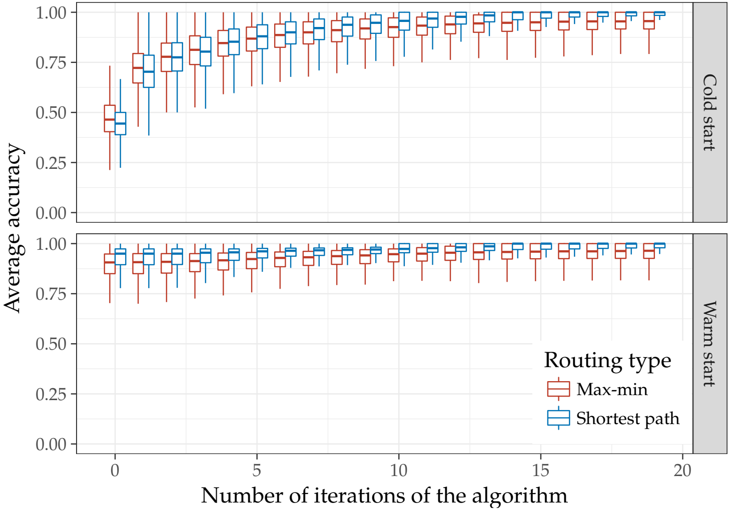
Accuracy of predicted routes

In average, 98% accuracy for shortest-path, 95% for max-min



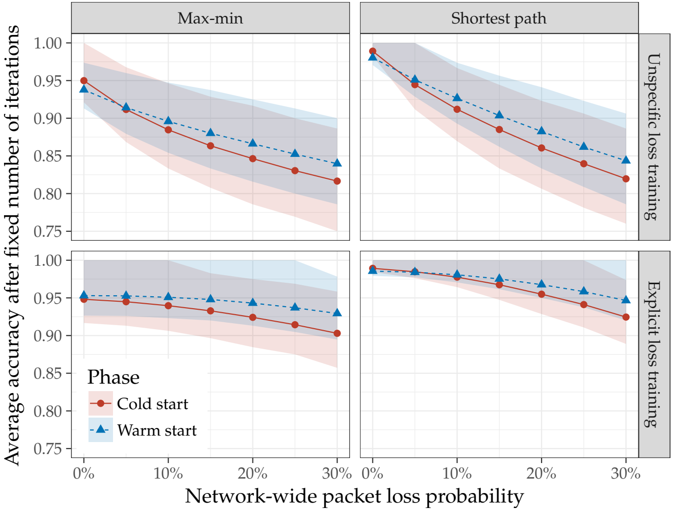
Numerical evaluation

Convergence time



Numerical evaluation

Resilience to packet loss



Numerical evaluation

Visualization of the protocol evolution

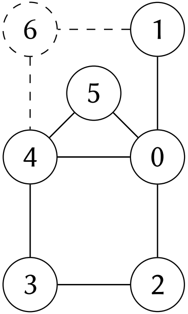
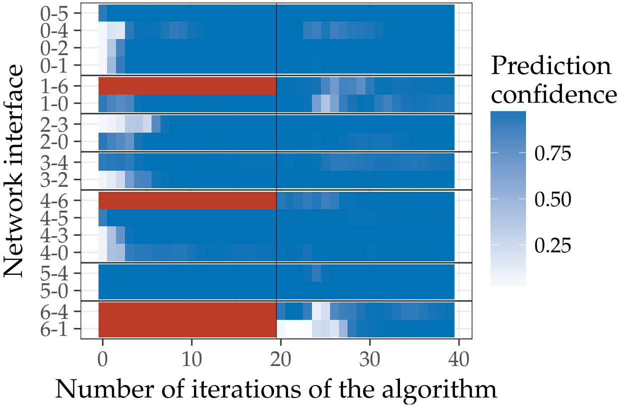


Figure 13: Evaluated topology. Node 6 is first offline and booted at iteration 20.



Conclusion

Summary

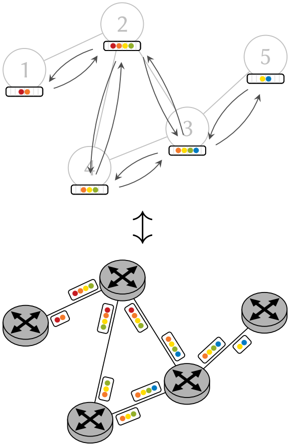
- **Generation of distributed routing protocol using Graph Neural Networks**
- Representation of network topologies as graphs
- Evaluations show that specific protocol properties can be explicitly trained

Key lesson

- Graph Neural Networks are well suited for reasoning about computer networks
- Also been applied to predict bandwidth [Geyer, 2017] and latency of protocols

Future work

- Comparison with manually-engineered protocols
- Generation of other distributed protocols



Bibliography

[Geyer, 2017] Geyer, F. (2017).
Performance Evaluation of Network Topologies using Graph-Based Deep Learning.
In *Proceedings of the 11th International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS 2017, pages 20–27.

[Knight et al., 2011] Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., and Roughan, M. (2011).
The Internet Topology Zoo.
IEEE J. Sel. Areas Commun., 29(9):1765–1775.

[Li et al., 2016] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2016).
Gated Graph Sequence Neural Networks.
In *Proceedings of the 4th International Conference on Learning Representations*, ICLR'2016.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009).
The Graph Neural Network Model.
IEEE Trans. Neural Netw., 20(1):61–80.

[Veličković et al., 2018] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018).
Graph Attention Networks.
In *International Conference on Learning Representations*.

Backup

Convergence time

Comparison with theoretical protocol based on graph diameter