DeepMPLS: Fast Analysis of MPLS Configurations Using Deep Learning

Fabien Geyer[1,2] and Stefan Schmid[3]

IFIP Networking 2019

Tuesday 21st May, 2019

[1] Chair of Network Architectures and Services
Technical University of Munich (TUM)

[2] Airbus Central R&T
Munich

[3] Faculty of Computer Science,
University of Vienna, Austria

Motivation

Network failures can have a large impact

- Github: *We discovered a misconfiguration on this pair of switches that caused what's called a "**bridge loop**" in the network*
- Amazon: *A network change was [...] executed incorrectly [...] more "stuck" volumes and added more requests to the **re-mirroring storm***
- GoDaddy: *Service outage was due to a series of internal network events that **corrupted router data tables**.*
- United Airlines: *Experienced a **network connectivity issue** [...] interrupted the airline's flight departures, airport processing and reservations systems*

Managing network is hard

- Mostly done by human with limited automation
- **Can we provide better tools and methods for assisting sysadmins?**

Motivation

Network automation and verification

Challenges in routing

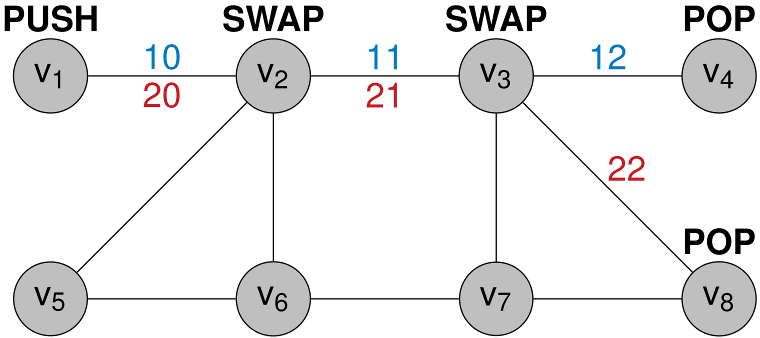
- Reachability: Can traffic from ingress port A reach egress port B?
- Loop-freedom: Are the routes implied by the forwarding rules loop-free?
- Policy: Is it ensured that traffic from A to B never goes via C?
- Waypoint enforcement: Is it ensured that traffic from A to B is always routed via a node C (e.g., intrusion detection system or a firewall)?

Automation and formal verification

- Some routing properties can be formally verified . . .
- . . . but it comes at a computational cost and leaves routing configuration to sysadmin

Motivation

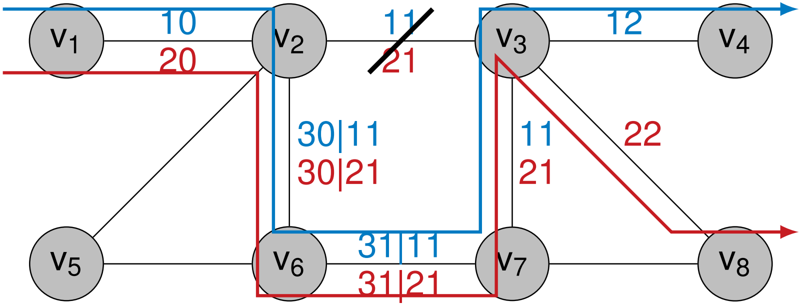
Analysis of MPLS networks – Example network



MPLS Configuration

Motivation

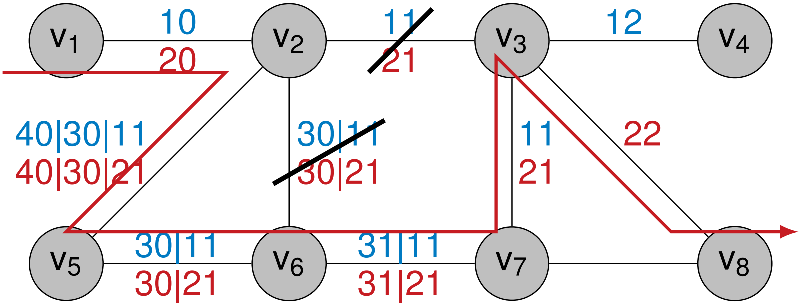
Analysis of MPLS networks – Example network



Fast Reroute Around 1 Failure

Motivation

Analysis of MPLS networks – Example network



Fast Rerouting may lead to inefficient paths

Motivation

Automated analysis of MPLS configuration

Formal verification

- Related work: NetKAT [Anderson et al., 2014], HSA [Kazemian et al., 2012], VeriFlow [Khurshid et al., 2013], Anteater [Mai et al., 2011]
- Difficult problem: some existing tools have a super-polynomial runtime, some verification are even undecidable

Polynomial-time solution

- Proposal using Push-Down Automata to verify MPLS networks [Schmid and Srba, 2018]
- P-Rex tool available [Jensen et al., 2018]
- Validation of MPLS queries using regular expressions in the form of: $< a > b < c > k$
- Only allows to *detect* but not *fix* configurations

Motivation

Deep Learning

Challenges

- **Can we speed-up the network verification?**
- **What about fixing and optimizing network configurations?**

General idea

- **Build a framework for combining analysis of MPLS networks and deep learning**
- Model problem as graph and process the graph using neural networks
- Predictions of the neural network can be used to statistically infer properties of the network

Outline

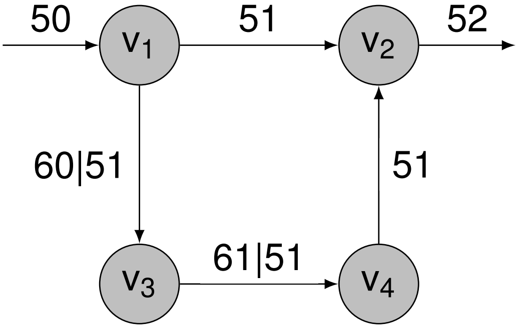
Graph Neural Network

Numerical evaluation

Conclusion

Graph Neural Network

Graph encoding - Network and MPLS configuration



Nodes

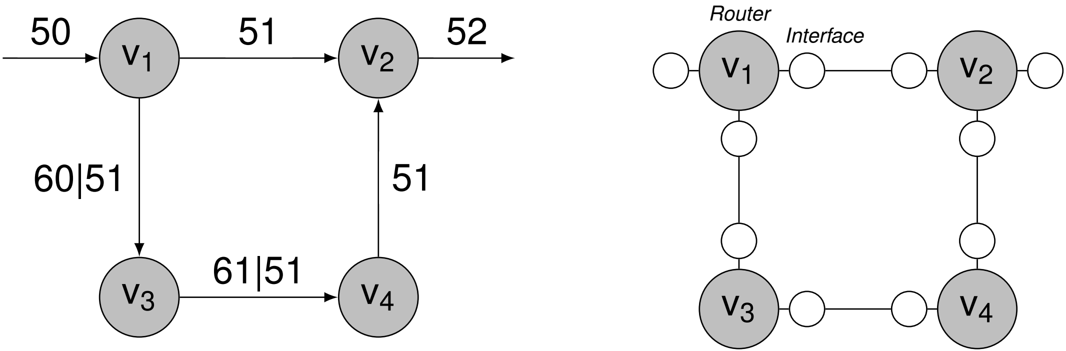
- **Physical network**: routers and interfaces
- **MPLS elements**: Rules, labels, actions
- **Query** and elements of regex

Edges

- Relationship between nodes

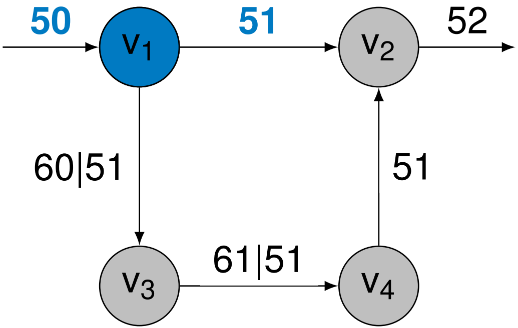
Graph Neural Network

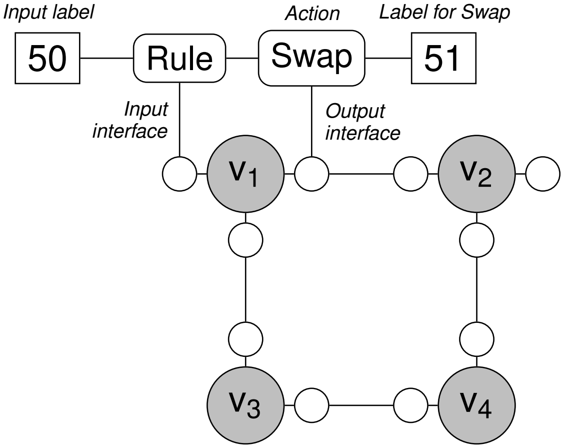
Graph encoding - Network and MPLS configuration



Graph Neural Network

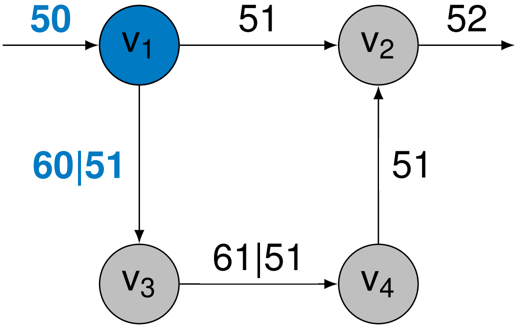
Graph encoding - Network and MPLS configuration

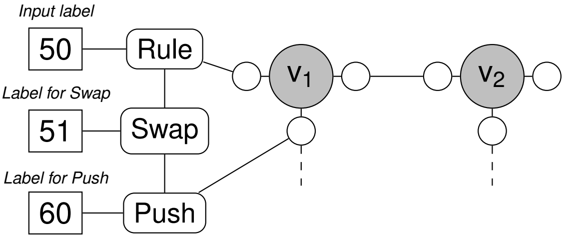




Graph Neural Network

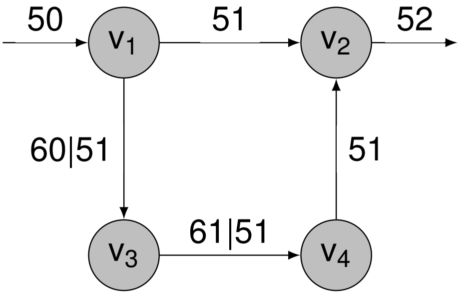
Graph encoding - Network and MPLS configuration



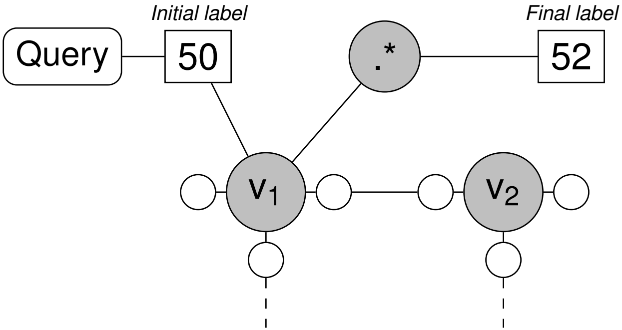


Graph Neural Network

Graph encoding - Query

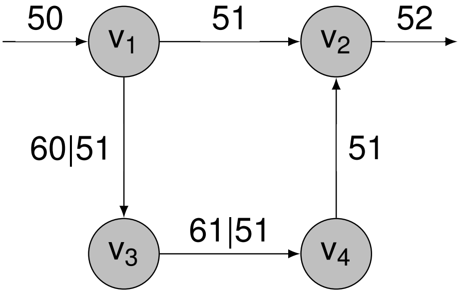



```
<50> v1 .* <52>
```

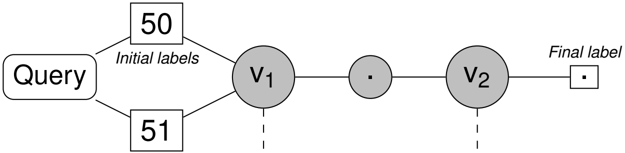


Graph Neural Network

Graph encoding - Query



`<50+51> v1 . v2 <.>`



Graph Neural Network

Graph encoding - Node features

Input features

- Node type encoded as categorical feature
- Edges have no input feature

Output features

- Binary classification problem for some nodes

Predictions

- Satisfiability Heuristic for verifying if a query is satisfiable
- Routing trace Heuristic for generating a trace of routers which match a satisfiable query
- Partial synthesis Synthesis of an MPLS configuration in order to satisfy a query

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

Principle

- Each node has a *hidden* vector $\mathbf{h}_v \in \mathbb{R}^k$
- ...computed according to the vector of its neighbors
- ...and are propagated through the graph

Algorithm

- Initialize $\mathbf{h}_v^{(0)}$ according to features of nodes
- for $t = 1, \ldots, T$ do
  - $\mathbf{a}_v^{(t)} = AGGREGATE\left(\left\{\mathbf{h}_u^{(t-1)} \mid u \in Nbr(v)\right\}\right)$
  - $\mathbf{h}_v^{(t)} = COMBINE\left(\mathbf{h}_v^{(t-1)}, \mathbf{a}_v^{(t)}\right)$
- return $READOUT\left(\mathbf{h}_v^{(T)}\right)$

Implementation (simplified)

- Initialize $\mathbf{h}_v^{(0)}$ according to features of nodes
- for $t = 1, \dots, T$ do
  - *AGGREGATE* $\rightarrow \mathbf{a}_v^{(t)} = \sum_{u \in Nbr(v)} \mathbf{h}_u^{(t-1)}$
  - *COMBINE* $\rightarrow \mathbf{h}_v^{(t)} = \textit{Neural Network}\left(\mathbf{h}_v^{(t-1)}, \mathbf{a}_v^{(t)}\right)$
- *READOUT* $\rightarrow$ return *Neural Network* $\left(\mathbf{h}_v^{(T)}\right)$

Training

- Using standard gradient descent techniques

Different approaches

- **Gated-Graph Neural Network**
- Graph Convolution Network
- Graph Attention Networks
- Graph Spatial-Temporal Networks
- ...

$\rightarrow$ Hot area of research in the ML community

Numerical evaluation

Dataset generation

- Generation of more than 90.000 topologies based on the Network Zoo [Knight et al., 2011]
- Generation of MPLS rules and queries based on random generator
- Validation of the MPLS configurations using P-Rex [Jensen et al., 2018]
- Dataset available online: `https://github.com/fabgeyer/dataset-networking2019`

Parameter	Min	Max	Mean	Median
# of routers	3	30	10.6	10
# MPLS labels	8	689	225.3	174
# MPLS rules	8	795	319.5	248
Size of push-down automaton	17	37006	5441.2	2692
# of nodes in analyzed graph	36	2333	914.4	713
# of edges in analyzed graph	48	4000	1615.4	1261

Table 1: Statistics about the generated dataset.

Types of queries:

- $< l_i > r_i < l_o > k$
- $< l_i > r_i .^* r_o < l_o > k$
- $< l_i > . .^* r_o < l_o > k$
- $< .^* > r_i .^* r_o < l_o > k$
- $< l_i > r_i .^* r_o < .^* > k$

Numerical evaluation

Baselines

Reminder on tasks

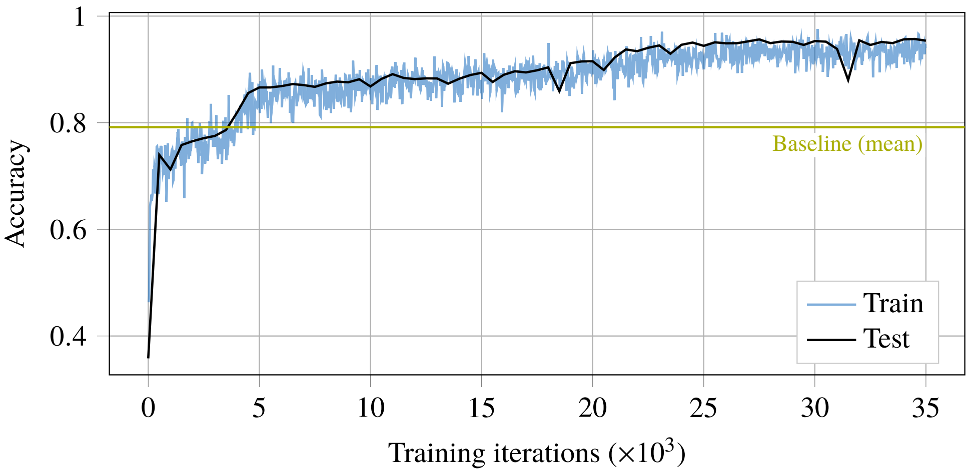
Satisfiability	Heuristic for verifying if a query is satisfiable
Routing trace	Heuristic for generating a trace of routers which match a satisfiable query
Partial synthesis	Synthesis of an MPLS configuration in order to satisfy a query

Comparison between machine learning results with a random-based baseline

- For the Satisfiability and Routing trace tasks: random walk in the MPLS network
- For the Partial synthesis task: random choice

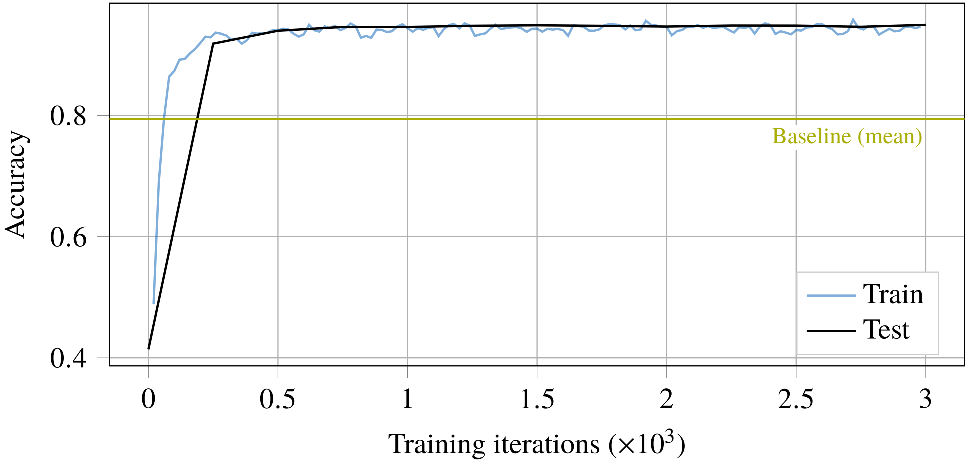
Numerical evaluation

Query satisfiability - Neural Network Training



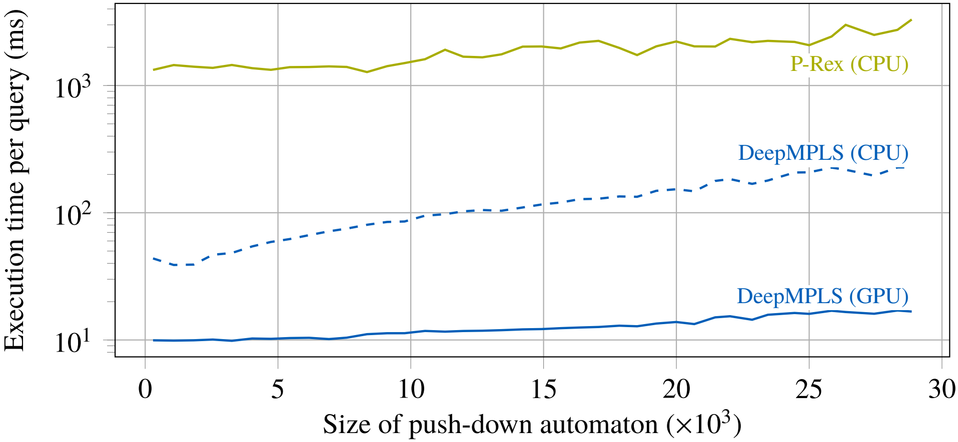
Numerical evaluation

Routing trace - Neural Network Training



Numerical evaluation

Runtime



Conclusion

Contributions

- **Framework combining MPLS analysis and graph-based deep learning**
- **Fast heuristic for verifying MPLS configurations**
- **Prediction of actions to take to fix MPLS configurations**
- First steps towards more complicated tasks and networks
- Dataset: `https://github.com/fabgeyer/dataset-networking2019`

Future work

- Synthesis of full MPLS configurations based on reinforcement learning
- Test and generalize our approach for other configurations, e.g., based on Segment Routing

[Anderson et al., 2014] Anderson, C. J., Foster, N., Guha, A., Jeannin, J.-B., Kozen, D., Schlesinger, C., and Walker, D. (2014).
Netkat: Semantic foundations for networks.
SIGPLAN Not., 49(1).

[Jensen et al., 2018] Jensen, J. S., Krogh, T. B., Madsen, J. S., Schmid, S., Srba, J., and Thorgersen, M. T. (2018).
P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures.
In *Proc. 14th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*.

[Kazemian et al., 2012] Kazemian, P., Varghese, G., and McKeown, N. (2012).
Header space analysis: Static checking for networks.
In *Proc. of USENIX NSDI*.

[Khurshid et al., 2013] Khurshid, A., Zou, X., Zhou, W., Caesar, M., and Godfrey, P. B. (2013).
Veriflow: verifying network-wide invariants in real time.
In *Proc. of USENIX NSDI*, pages 15–27.

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

[Mai et al., 2011] Mai, H., Khurshid, A., Agarwal, R., Caesar, M., Godfrey, P., and King, S. T. (2011).
Debugging the data plane with anteater.
In *ACM SIGCOMM Computer Communication Review*, volume 41 (4), pages 290–301.

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

[Schmid and Srba, 2018] Schmid, S. and Srba, J. (2018).
Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks.
In *Proc. of IEEE INFOCOM*.