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

PUSH

10

20

SWAP

11

21

SWAP

12

POP

What happens in case of link failure?

Fast Reroute Around 1 Failure

Fast Reroute Around 2 Failures

Fast Rerouting may lead to inefficient paths

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

Input label

Rule

Swap

Label for Swap

Input interface

Output interface

Graph Neural Network
Graph encoding - Network and MPLS configuration
Graph Neural Network

Graph encoding - Query

\[
\begin{align*}
\text{Initial label} & \quad \text{Final label} \\
\text{Query} & \quad \text{V1} \\
\text{50} & \quad \text{52} \\
\text{V1} & \quad \text{V2} \\
\text{V3} & \quad \text{V4} \\
\end{align*}
\]
Graph Neural Network
Graph encoding - Query
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 Network

Graph Neural Networks – Introduction

**Graph Neural Networks** [Scarselli et al., 2009] and related architectures are able to process general graphs and predict feature of nodes $o_v$

**Principle**

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

**Algorithm**

- Initialize $h_v^{(0)}$ according to features of nodes
- for $t = 1, \ldots, T$ do
  - $a_v^{(t)} = AGGREGATE \left( \{ h_u^{(t-1)} \mid u \in Nbr(v) \} \right)$
  - $h_v^{(t)} = COMBINE \left( h_v^{(t-1)}, a_v^{(t)} \right)$
- return $READOUT \left( h_v^{(T)} \right)$
Graph Neural Network
Graph Neural Networks – Implementation

Implementation (simplified)

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

Training

- Using standard gradient descent techniques

Different approaches

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

→ 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

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

Table 1: Statistics about the generated dataset.

Types of queries:

- $< l_i \geq r_i < l_o \geq k$
- $< l_i \geq r_i \cdot < r_o < l_o \geq k$
- $< l_i > . \cdot < r_o < l_o > k$
- $< r_i \cdot \cdot < r_o < \cdot > k$
- $< l_i > r_i \cdot < r_o < \cdot > k$
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

Baseline (mean)

Accuracy

Train
Test

Training iterations ($\times 10^3$)
Numerical evaluation

Routing trace - Neural Network Training

![Graph showing accuracy vs. training iterations](image)

- **Baseline (mean)**
- **Train**
- **Test**

<table>
<thead>
<tr>
<th>Training iterations ($\times 10^3$)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Numerical evaluation

Runtime

![Graph showing execution time per query (ms) vs. size of push-down automaton (×10^3). The graph compares P-Rex (CPU), DeepMPLS (CPU), and DeepMPLS (GPU).]
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
SIGPLAN Not., 49(1).

P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures.
In Proc. 14th International Conference on emerging Networking EXperiments and Technologies (CoNEXT).

Header space analysis: Static checking for networks.
In Proc. of USENIX NSDI.

Veriflow: verifying network-wide invariants in real time.

The Internet Topology Zoo.
IEEE Journal on Selected Areas in Communications, 29(9):1765–1775.

Debugging the data plane with anteater.

The Graph Neural Network Model.

Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks.
In Proc. of IEEE INFOCOM.