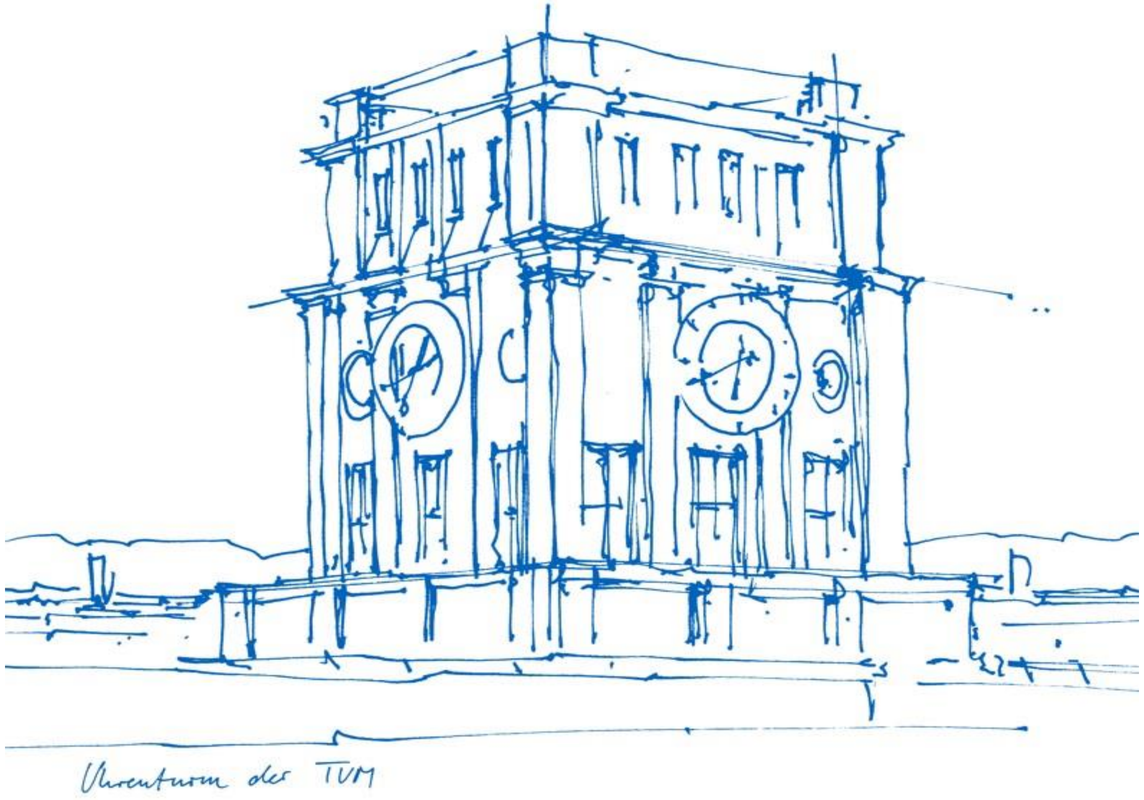
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technical University of Munich

From SDN Network Hypervisor Measurements to Fast Virtual Network Provisioning: A Long Story Short

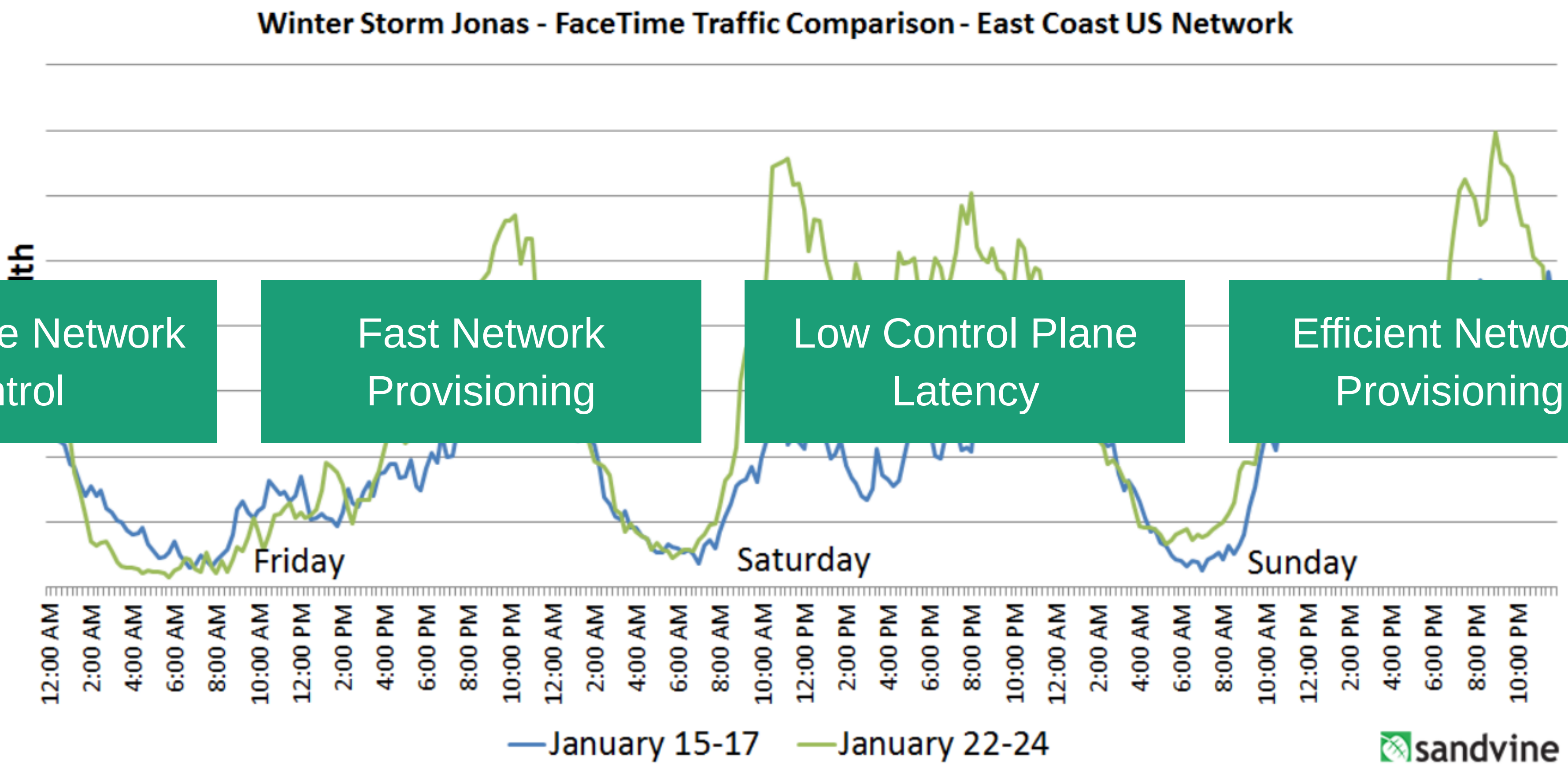
Andreas Blenk

**Technical University of Munich, Germany*

University of Vienna, 2018-10-10



Motivation for Flexibility



Predictable Network Control

Fast Network Provisioning

Low Control Plane Latency

Efficient Network Provisioning

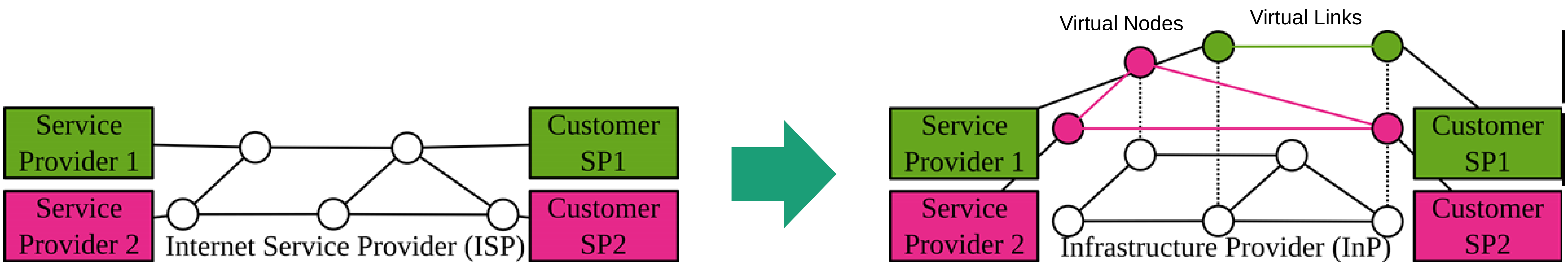
Frequently changing demands need flexible adaptation

Promising technologies and techniques:

Network Virtualization (NV), **Software-Defined Networking (SDN)**, **Artificial Intelligence (AI)**

Simply Combining Technologies and Still Predictable?

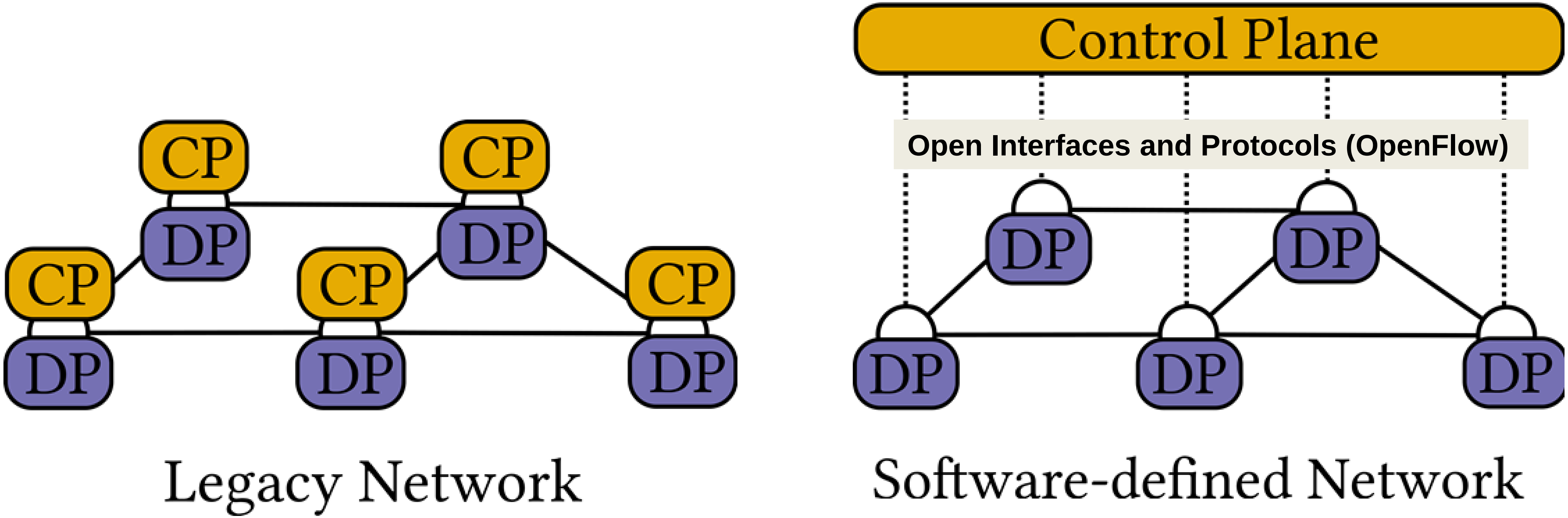
Network Virtualization



Flexible? Adaptive? Programmable?

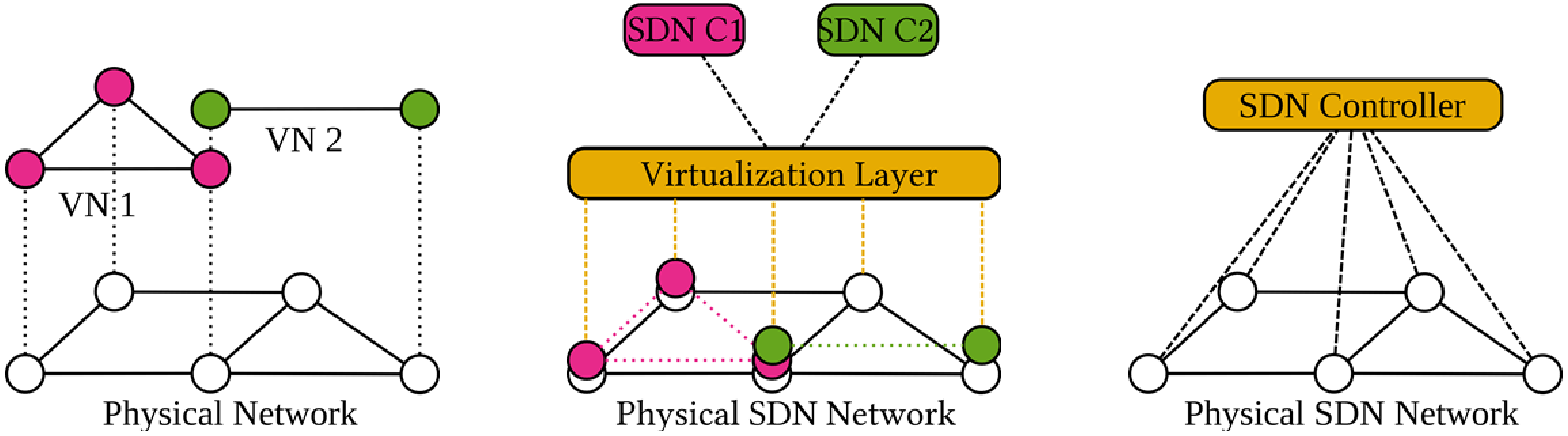
Combine with Software-Defined Networking

Software-Defined Networking



- Split control from data plane
- Centralized control
- Flexible control of forwarding (networking) resources

Combining Network Virtualization and Software-Defined Networking



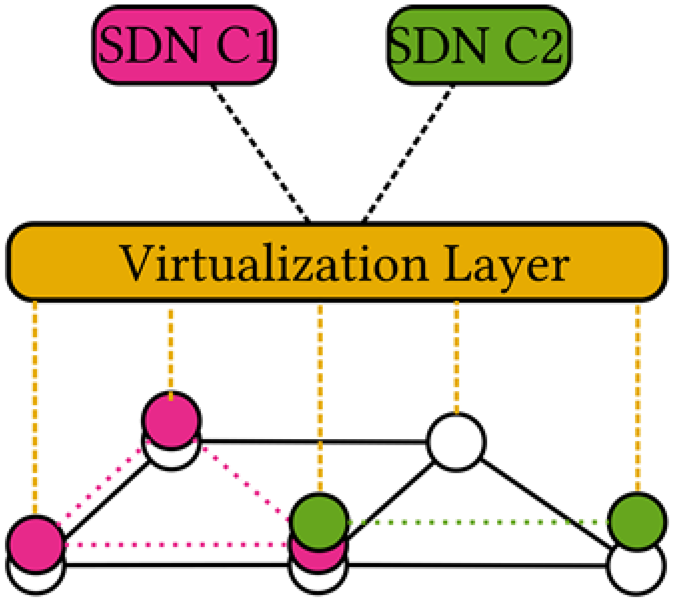
- Virtual Networks according to service and application demands
- Flexible control of virtual networking resources

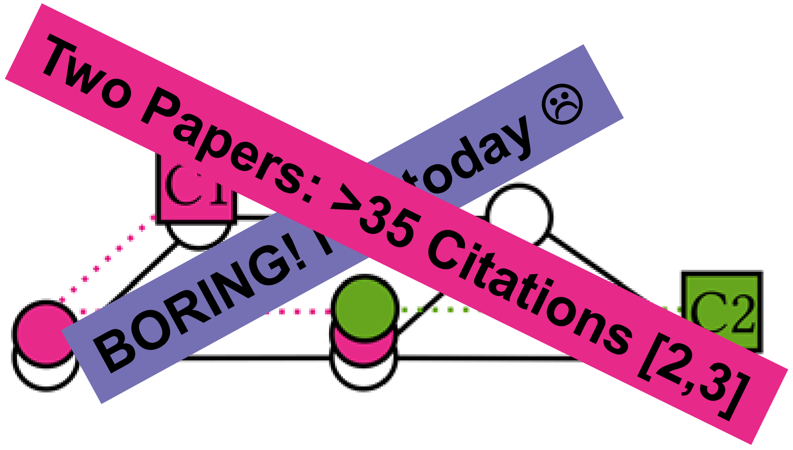
➔ Programmable virtual software-defined networks (vSDNs) [1]

And what is the problem now?

[1] A. Blenk, A. Basta, M. Reisslein, W. Kellerer, Survey on Network Virtualization Hypervisors for Software Defined Networking, IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 655-685, January 2016.

The Challenges





Architecture Design, Measurements

Function Placement, Optimization

- Virtualization itself can introduce overhead
- Interference due to sharing

➔Sources of unpredictability

➔Good understanding (models) of virtualization layer design needed for correct provisioning

[2] A. Blenk, A. Basta, J. Zerwas, W. Kellerer, Pairing SDN with Network Virtualization; The Hypervisor Placement Problem, IEEE NFV-SDN Conference, pp. 198-204, 2015

[3] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, W. Kellerer, Control Plane Latency with SDN Network Hypervisors: Cost of Virtualization, IEEE Transactions on Network and Service Management, September 2016

State of the art: Measurements and Models

FlowVisor: A Network Virtualization Layer

Rob Sherwood*, Glen Gibb†, Kok-Kiong Yap†, Guido Appenzeller†,
Martin Casado◇, Nick McKeown†, Guru Parulkar†
* Deutsche Telekom Inc. R&D Lab, Los Altos, CA USA
† Stanford University, Palo Alto, CA USA
◇ Nicira Networks, Palo Alto, CA USA

ABSTRACT

Network virtualization has long been a goal of of the network research community. With it, multiple isolated logical networks each with potentially different addressing and forwarding mechanisms can share the same physical infrastructure. Typically this is achieved by taking advantage of the flexibility of software (*e.g.* [20, 23]) or by duplicating components in (often specialized) hardware[19].

In this paper we present a new approach to switch virtualization in which the same hardware forwarding plane can be shared among multiple logical networks, each with distinct forwarding logic. We use this switch-level virtualization to build a research platform which allows multiple network experiments to run side-by-side with production traffic while still providing isolation and hardware forwarding speeds. We also show that this approach is compatible with commodity switching chipsets and does not require the use of programmable hardware such as FPGAs or network processors.

We build and deploy this virtualization platform on our own production network and demonstrate its use in practice by running five experiments simultaneously within a campus network. Further, we quantify the overhead of our approach and evaluate the completeness of the isolation between virtual slices.

1. INTRODUCTION

This paper explores how to virtualize a network, and describes a particular system that we prototyped - called FlowVisor - that we have deployed to *slice*[1] our own production network. Similar to computer virtualization [22, 1, 21, 17], network virtualization promises to improve resource allocation, permits operators to checkpoint their network before changes, and allows competing customers to share the same equipment in a controlled and isolated fashion. Critically, virtual networks also promise to provide a safe and realistic environment to deploy and evaluate experimental "clean slate" protocols in production networks.

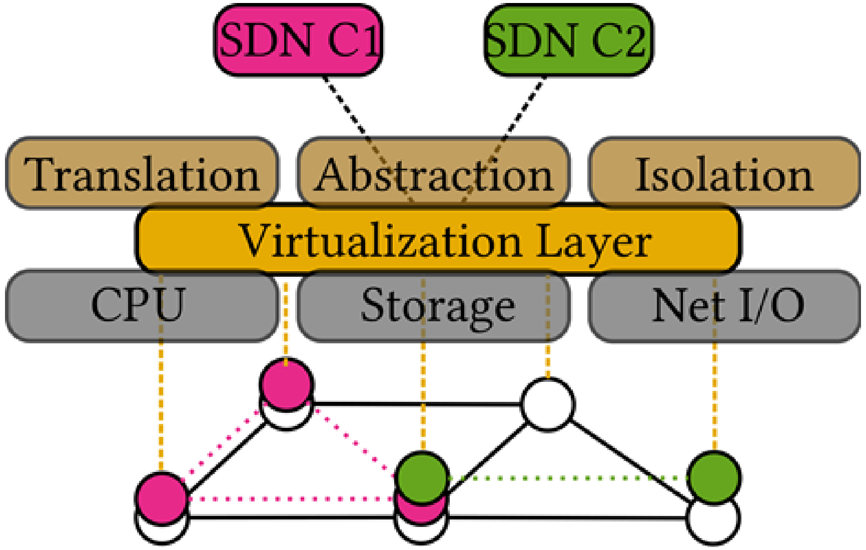
To better understand virtual networking, we first look closely at computer virtualization. Computer virtualization's success can be linked to a clean abstraction of the underlying hardware. That is, the computer virtualization layer has a hardware abstraction that permits slicing and sharing of resources among the guest operating systems. The effect is that each OS believes it has its own private hardware. A well defined hardware abstraction enables rapid innovation both above and below the virtualization layer. Above, the ability to build on a consistent hardware abstraction has allowed operating systems to flourish (e.g., UNIX, MacOS, several flavors of Linux, and Windows) and even encouraged entirely new approaches [24, 28]. Below, different hardware can be used (e.g., Intel, AMD, PPC, Arm, even Nvidia's GPU), so long as it can be mapped to the hardware abstraction layer. This allows different hardware to have different instruction sets optimized for higher performance, lower power, graphics, etc. Allowing choice above and below the virtualization layer means a proliferation of options, and a more competitive, innovative and efficient marketplace.

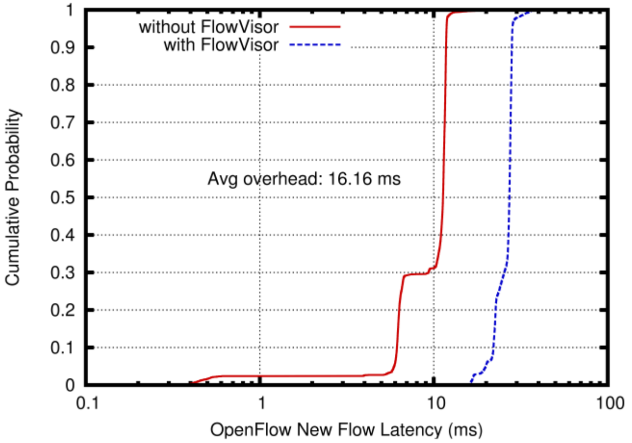
Our goal is to achieve the same benefits in the network. Thus, by analogy, the network itself should have a hardware abstraction layer. This layer should be easy to slice so that multiple wildly different networks can run simultaneously on top without interfering with each other, on a variety of different hardware, including switches, routers, access points, and so on. Above the hardware abstraction layer, we want new protocols and addressing formats to run independently in their own isolated slice of the same physical network, enabling networks optimized for the applications running on them, or customized for the operator who owns them. Below the virtualization layer, new hardware can be developed for different environments with different speed, media (wireline and wireless), power or fanout requirements.

The equipment currently deployed in our networks

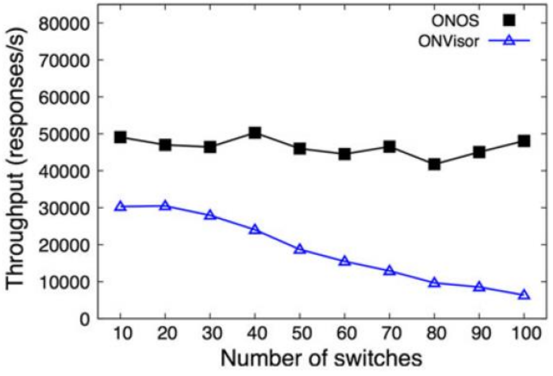
[1]Borrowing from the GENI [4] literature, we call an instance of a virtual network a *slice*, and two distinct virtual networks on the same physical hardware *slices*.

1





51 Packets/second! [FlowVisor]



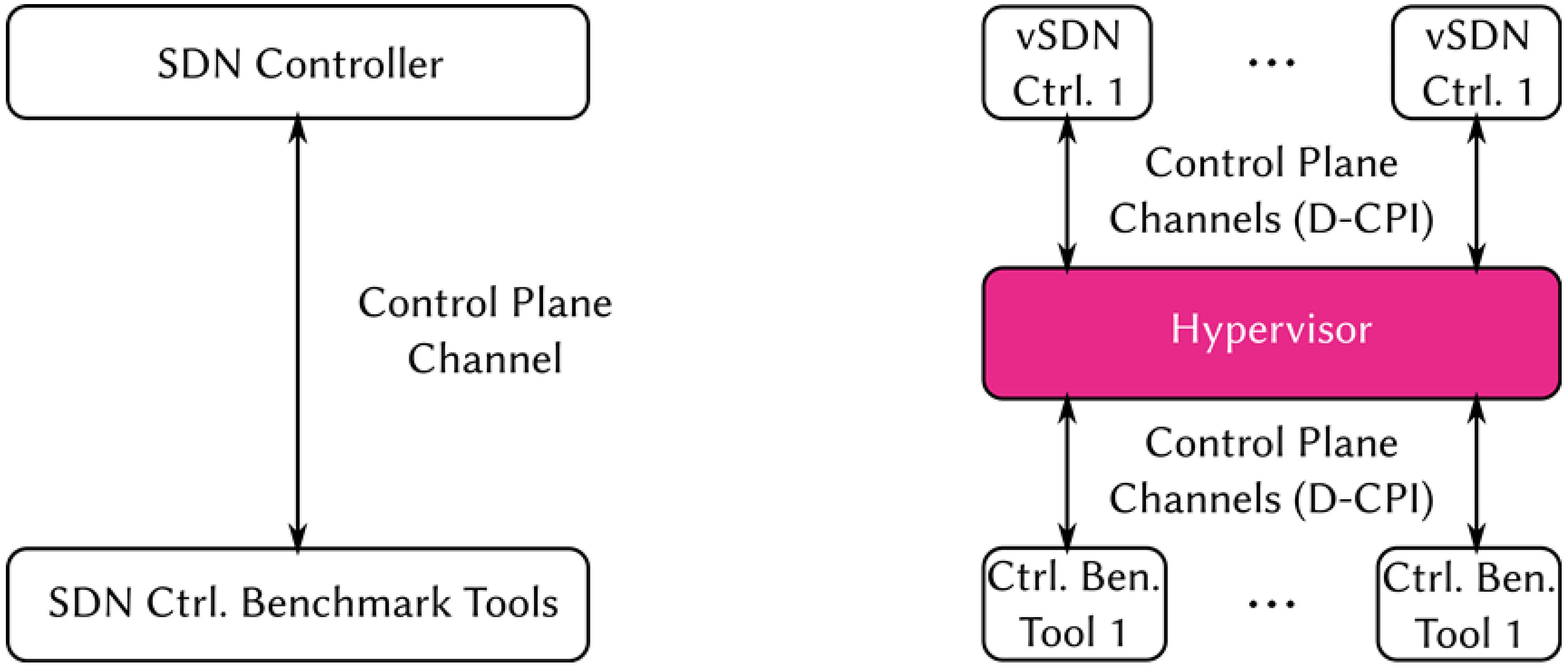
(B) Throughput comparison

1 Tenant only! [Onvisor2018]

No detailed performance study! Why? No Tool available!

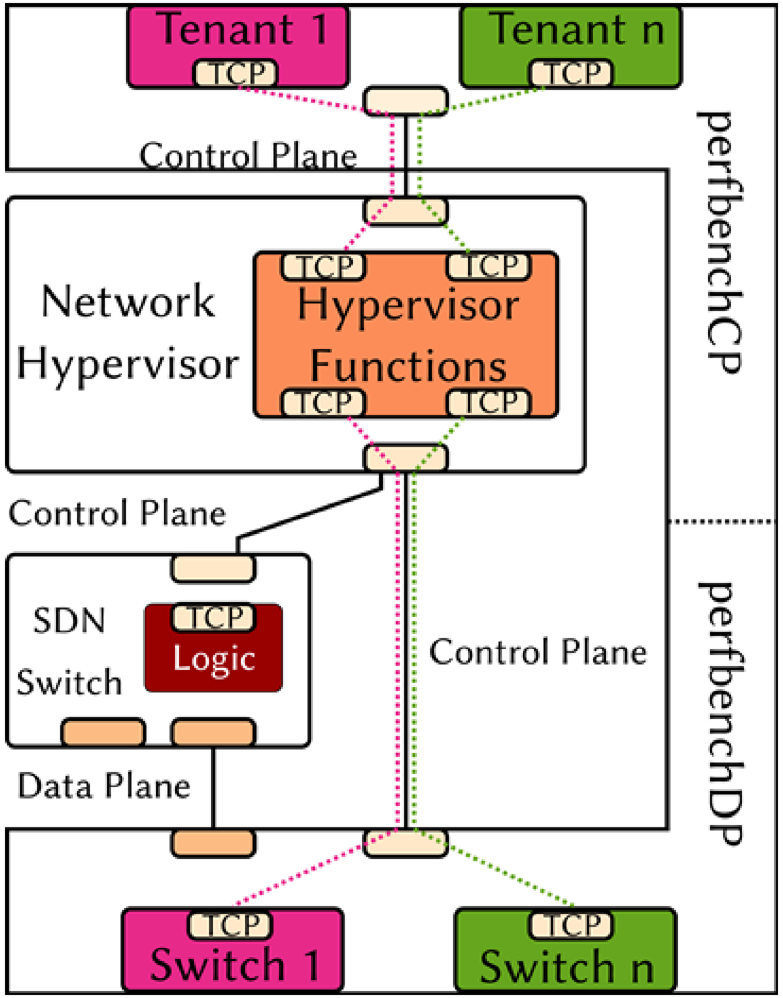
Measurement Procedure

From non-virtualized SDN networks to virtualized SDN networks



- Challenge: Coordination and emulation complexity
- Goal: One tool emulating single tenant, single switch, multi-tenant, multi-switch

Perfbench [4,5]



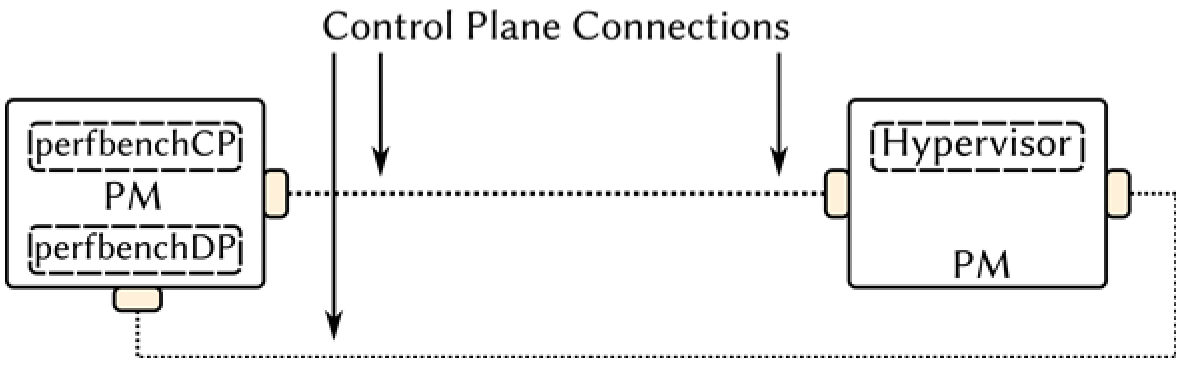
- Multi-tenant/multi-switch emulation
- Traffic modeling: inter-arrival time, burstiness
- Modular measurements: either controller(s), switch(es), or both entities

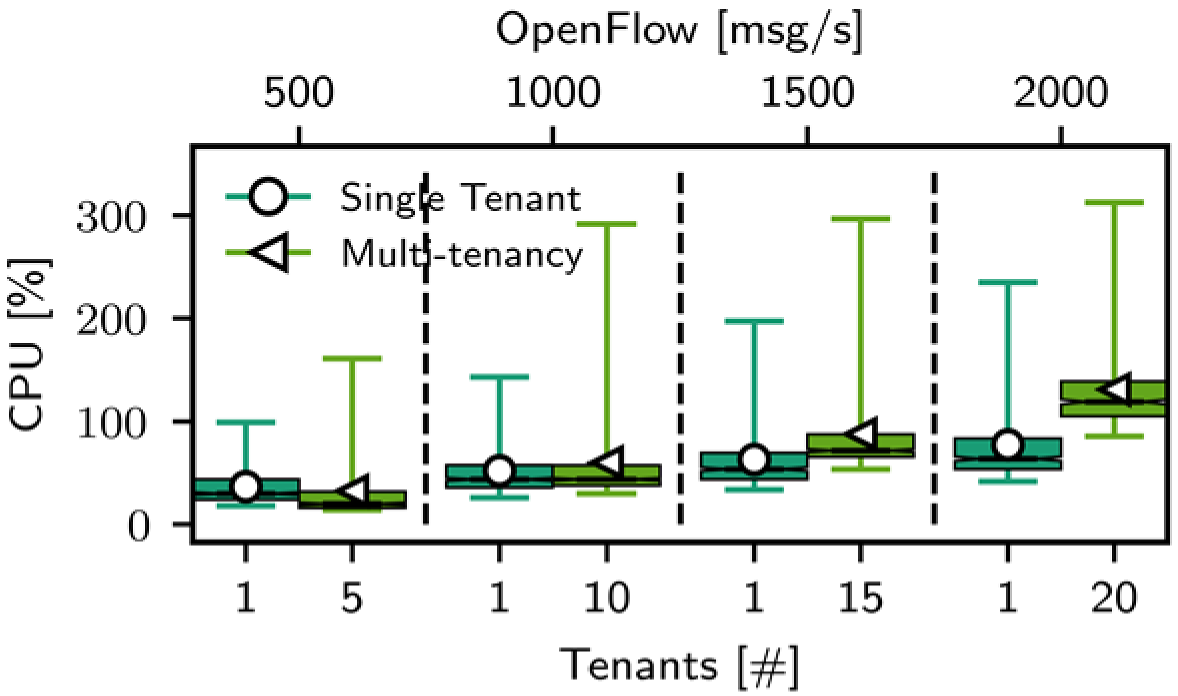
Open Source!

https://github.com/tum-lkn/perfbench

[4] A. Blenk, A. Basta, L. Henkel, J. Zerwas, S. Schmid, W. Kellerer, perfbench: A Tool for Predictability Analysis in Multi-Tenant Software Defined Networks. ACM SIGCOMM 2018 Conference Posters and Demos, 2018,
[5] A. Basta, A. Blenk, S. Dudycz, A. Ludwig, S. Schmid ,Efficient Loop-Free Rerouting of Multiple SDN Flows. IEEE/ACM Transactions on Networking 26 (2), 2018, pp. 948-961

Multi-Tenancy Measurement Setup

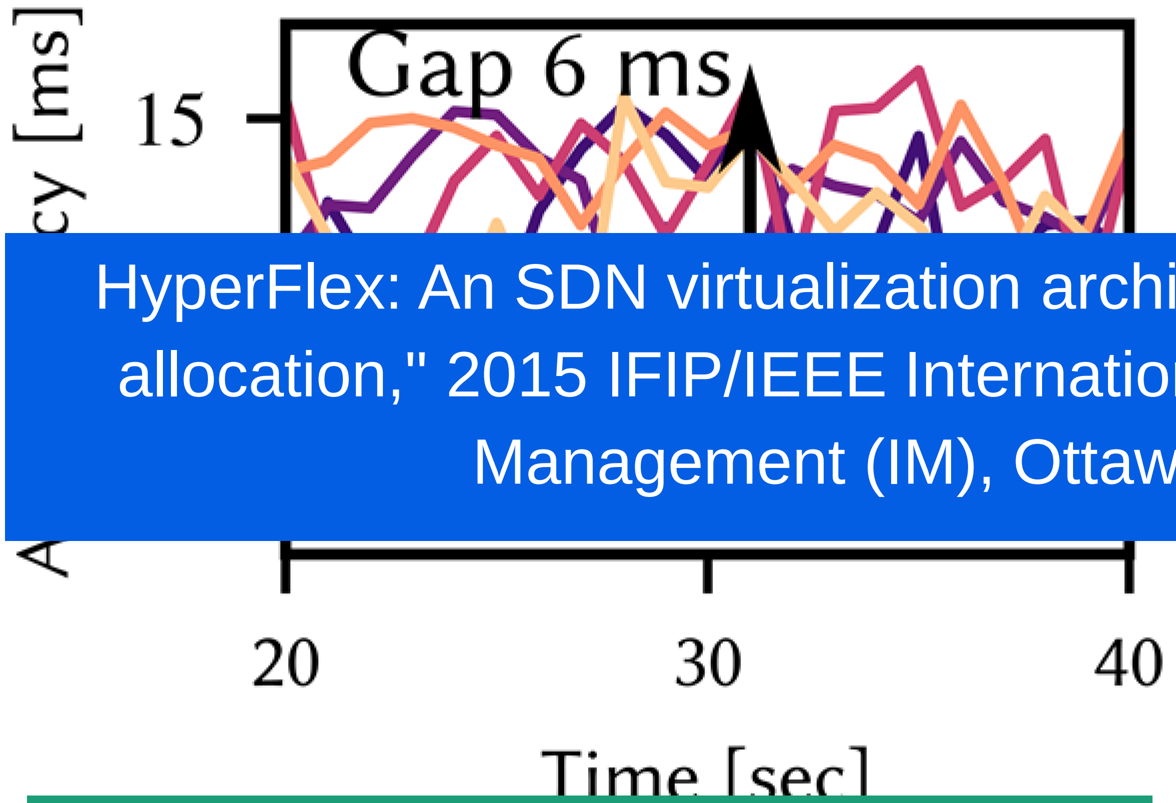


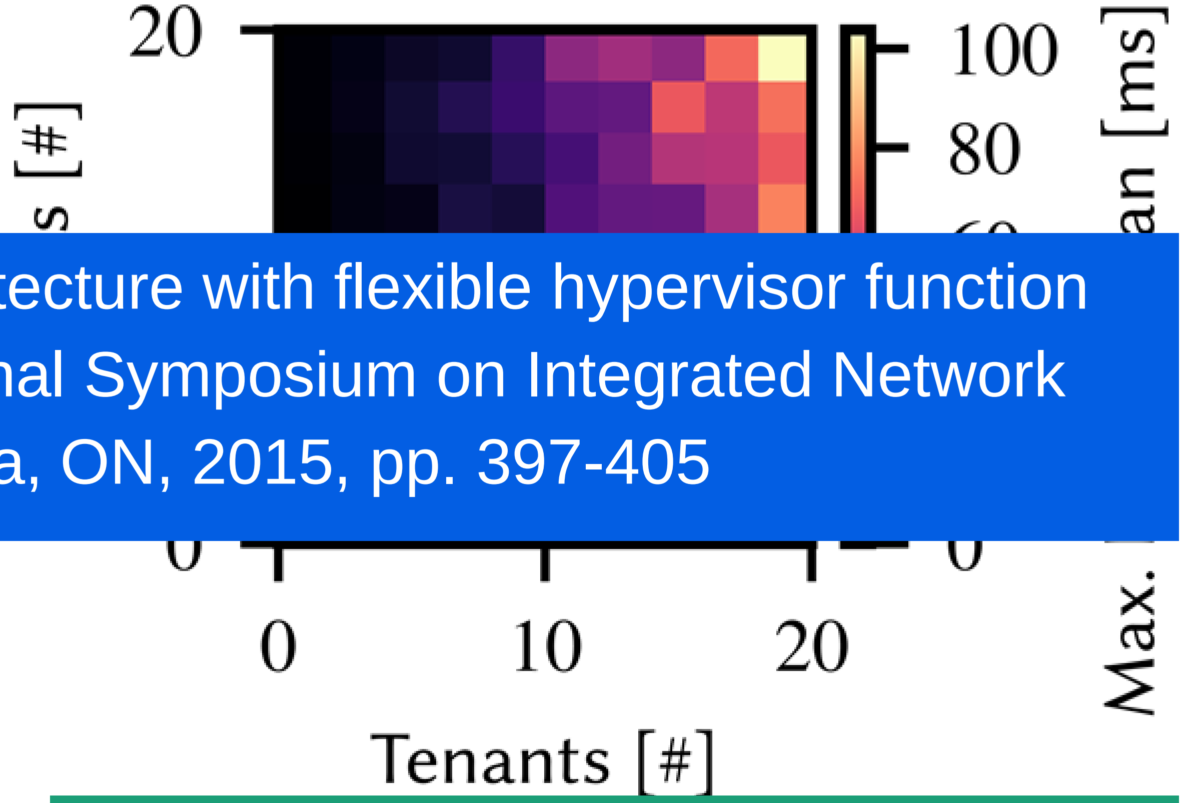


- Hypervisor: FlowVisor
- OpenFlow Message: FLOWMOD
- Key performance indicator:
  - Latency [milliseconds]
  - CPU [%] (100% = 1 Core)

- Multi-tenancy impact on CPU consumption
- Impact on control plane latency?

Multi-Tenancy Latency Results





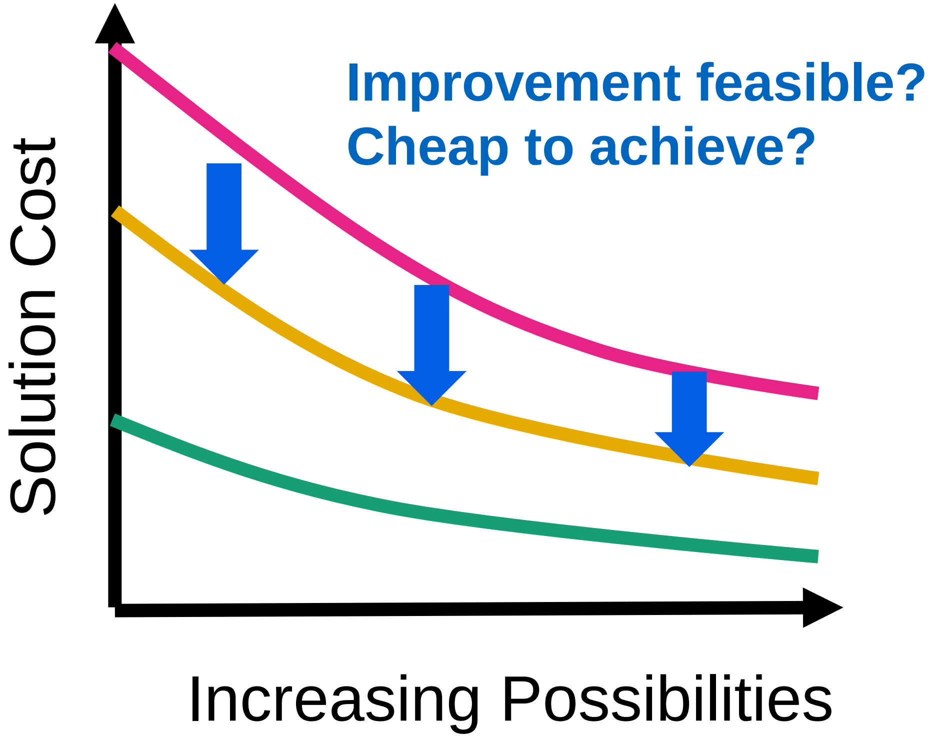
HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 397-405

Already 10 tenants show a notable latency gap of 6ms (high variance)

The more switches and controllers, the less predictable

Fast and Efficient (Virtual) Network Provisioning

New Opportunities Introduce New Problems and Challenges

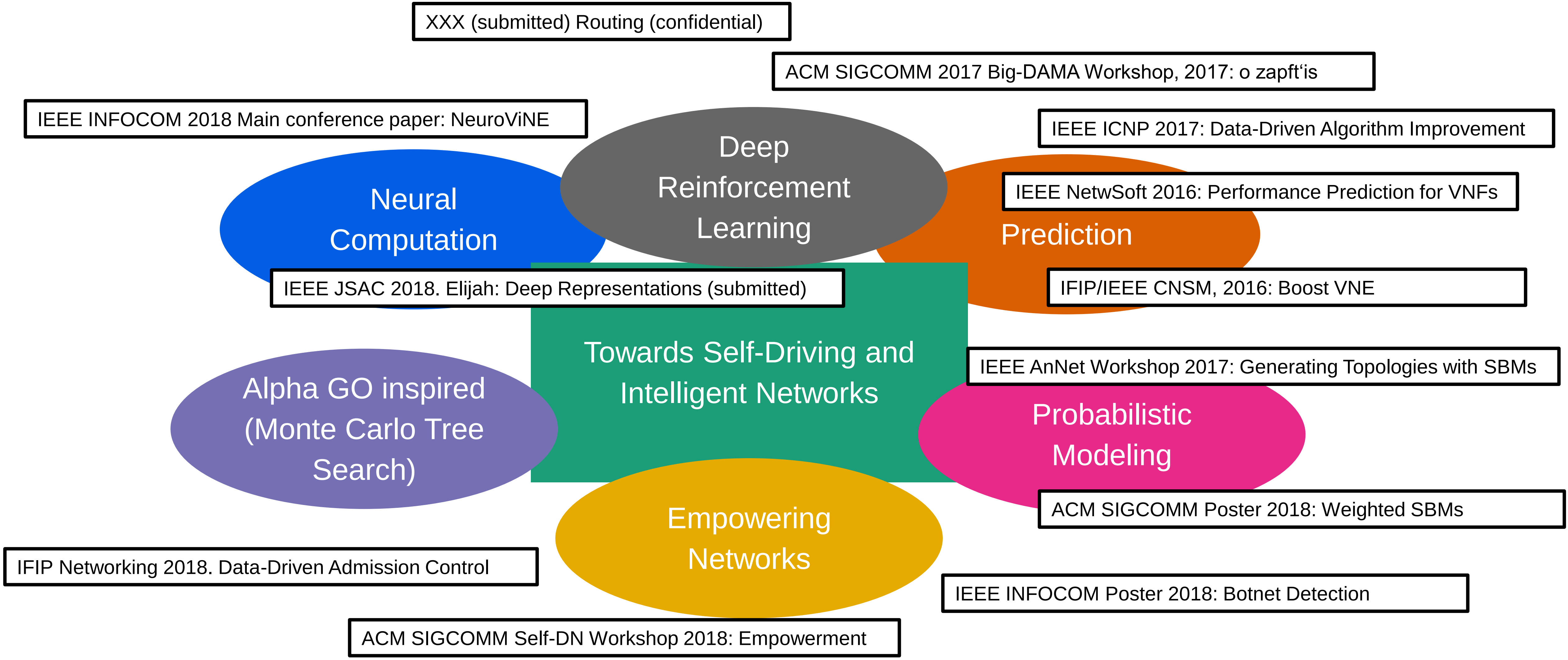


- **Optimal Algorithm:**
  - Improves solution quality given more flexibilities
  - Expensive, exponential runtime
- **Heuristic Algorithm:**
  - Can exploit flexibility
  - But cannot achieve optimal solution
- **Machine Learning/Neural Computation:**
  - Improves solution quality
  - Impact of learning time? Computational overhead? RESEARCH!

How some people see AI and Machine Learning!

If you work a bit with it …
it should be your friend and helper!

Machine Learning for Networking



Overview in this talk: NeuroViNE

ACM SIGCOMM 2017 Big-DAMA Workshop, 2017: o zapft‘is

IEEE INFOCOM 2018 Main conference paper: NeuroViNE

Neural Computation

Prediction

IFIP/IEEE CNSM, 2016: Boost VNE

Towards Self-Driving and Intelligent Networks

Probabilistic Modeling

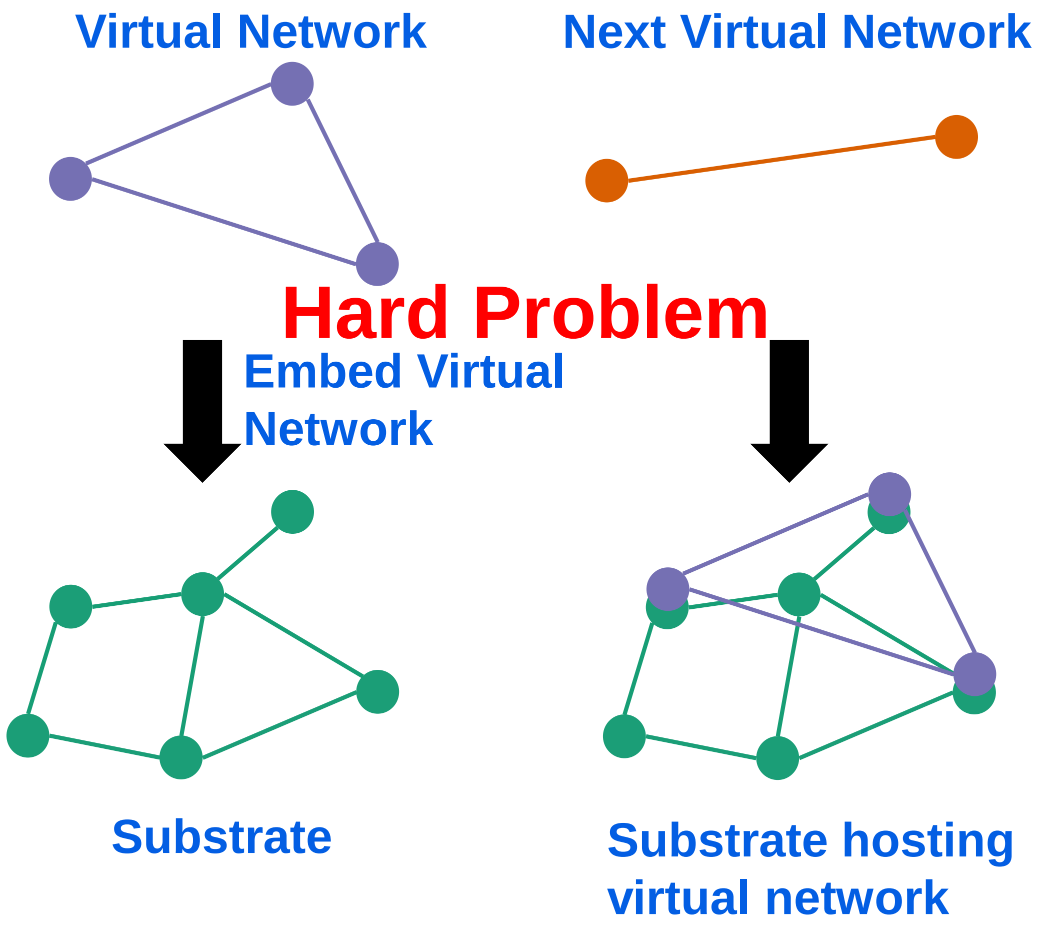
Empowering Networks

ACM SIGCOMM Poster 2018: Weighted SBMs

IEEE INFOCOM Poster 2018: Botnet Detection

ACM SIGCOMM Self-DN Workshop 2018: Empowerment

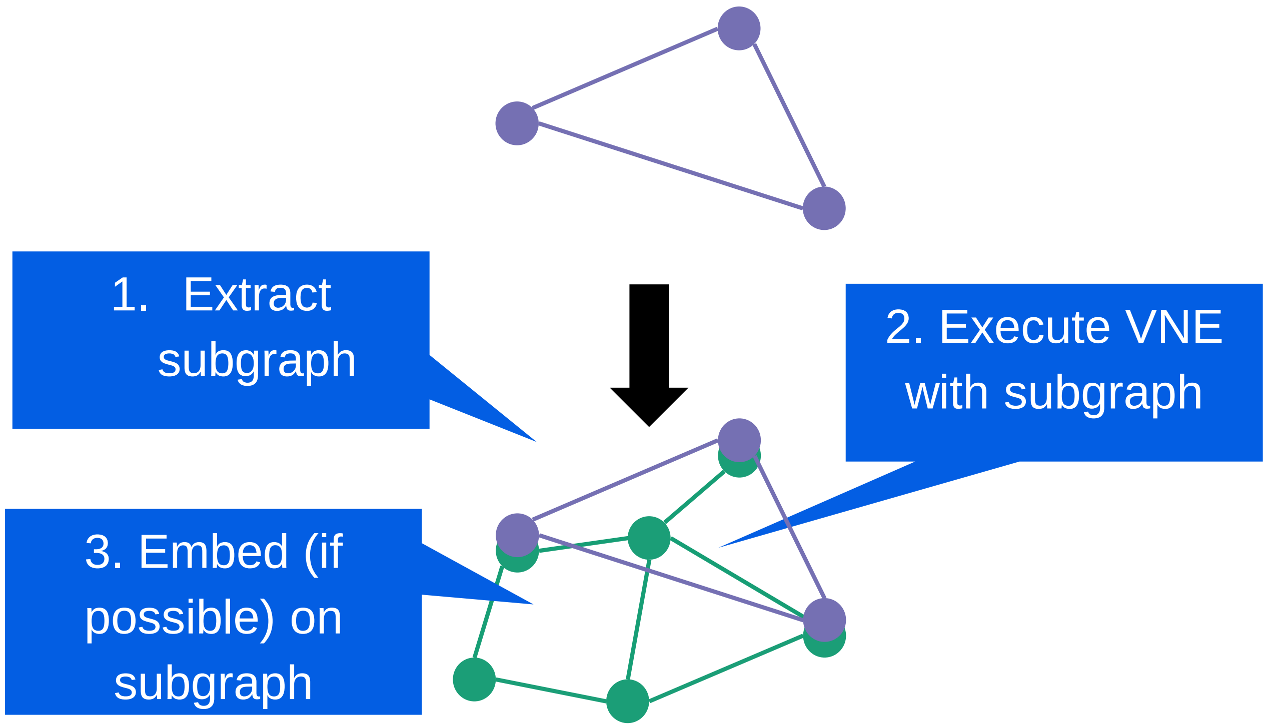
Use case study: Online Virtual Network Embedding (VNE) problem



(1) Optimal solutions do not scale
Vs.
(2) Heuristics may result in large footprints

Neural Preprocessing to achieve (1) scalability and (2) quality

The Idea: Subgraph Extraction



→Reduce embedding cost of heuristics (search on close substrate nodes)

→Improve runtime of optimal algorithms (shrink search space)

But how do we find good subgraphs?!

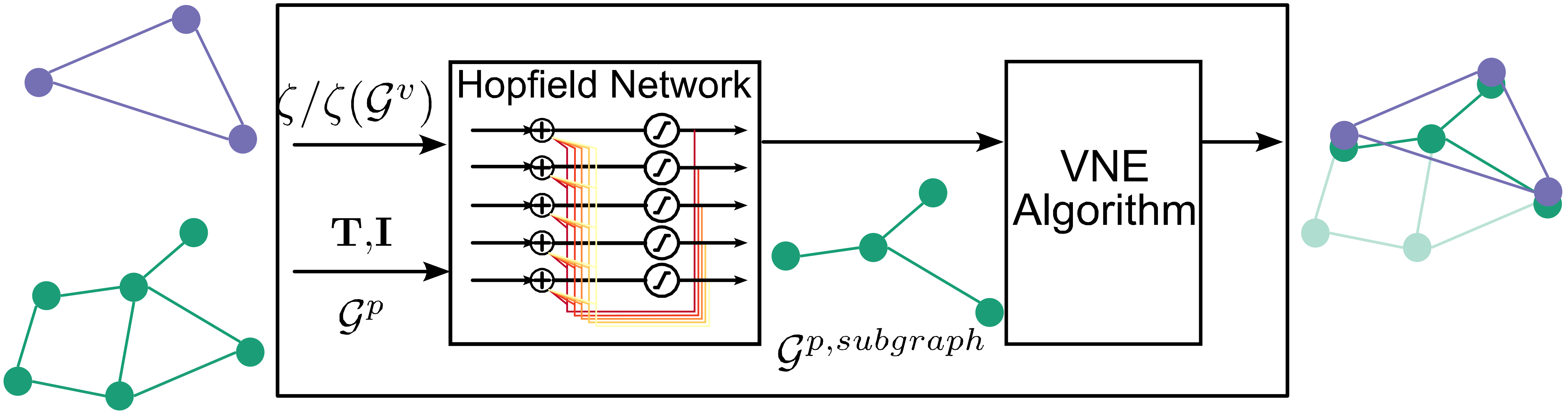
Contribution: NeuroViNE [6]

Neural Computation

Parallel Computation

Implementable on hardware

Reuse existing VNE algorithms



Hopfield network solution provides nodes with high capacity close to each other

[6] A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, W. Kellerer, NeuroViNE: A Neural Preprocessor for Your Virtual Network Embedding Algorithm, 37th IEEE Conference on Computer Communication (INFOCOM),pp. 405-413, 2018

Neural Computation of Decisions in Optimization Problems

“Neural” computation of decisions in optimization problems

JJ **Hopfield**, DW Tank - Biological cybernetics, 1985 - Springer

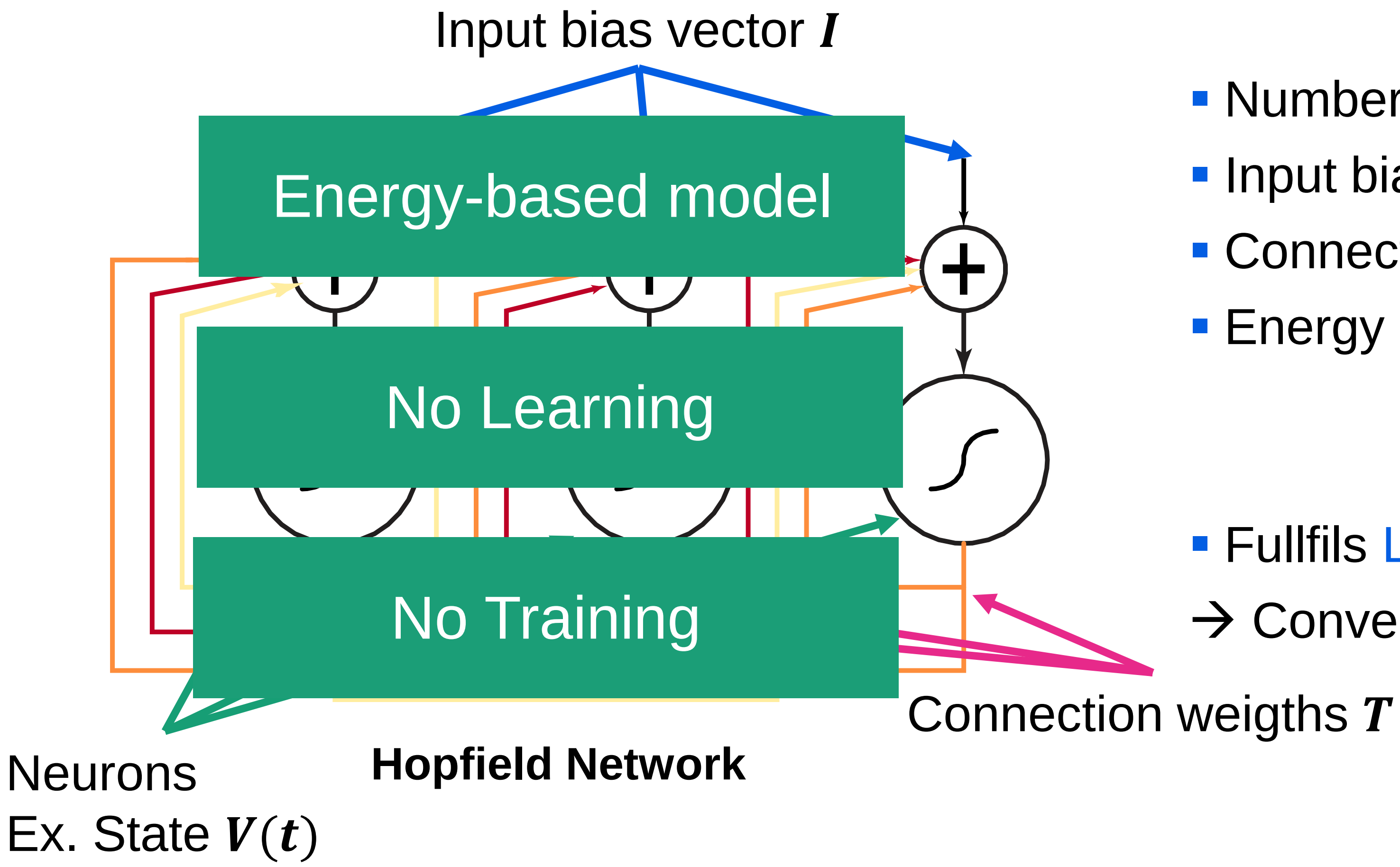
Abstract Highly-interconnected networks of nonlinear analog neurons are shown to be extremely effective in computing. The networks can rapidly provide a collectively-computed solution (a digital output) to a problem on the basis of analog input information. The …

☆ 99 Zitiert von: 7329 Ähnliche Artikel Alle 33 Versionen

John Hopfield

Hopfield Network

An Artificial Recurrent Neural Network (which can be used for optimization)



Hopfield Network

- Number of neurons and states $\boldsymbol{V}$
- Input bias vector $\boldsymbol{I}$
- Connection weigths $\boldsymbol{T}$
- Energy of network

$$E = -\frac{1}{2}\boldsymbol{V}^T\boldsymbol{T}\boldsymbol{V} - \boldsymbol{V}^T\boldsymbol{I}$$

- Fullfils Lyapunov function property

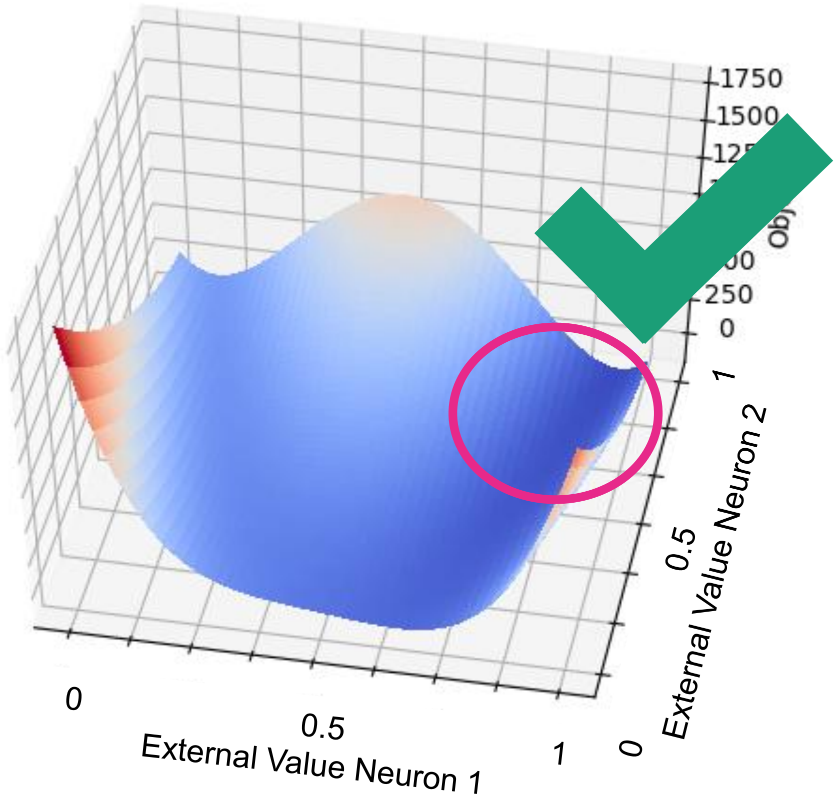
→ Convergence to local (global) optima guaranteed

Hopfield Network Properties

How to map Virtual Network Embedding problem?

Hopfield Network

How to use for optimization ...



1. Optimization problem: find subgraph with low resource footprint and high probability for accepting virtual network
2. VNE problem energy function

   $E = \boldsymbol{V}^T\left(\boldsymbol{\Psi}(\mathrm{t}) + \alpha \cdot \mathbf{T}^{\mathrm{constraint}}\right)\boldsymbol{V} + \boldsymbol{V}^T\left(\boldsymbol{\Xi}(\mathrm{t}) + \alpha \cdot \mathbf{I}^{\mathrm{constraint}}\right)$
3. Derive: $\boldsymbol{\Psi}(\mathrm{t})$, $\mathbf{T}^{\mathrm{constraint}}$, $\boldsymbol{\Xi}(\mathrm{t})$, $\mathbf{I}^{\mathrm{constraint}}$
4. Execute network: solve
5. After exectution → Neuron states (values) indicate subgraph nodes

Hopfield Optimization Procedure

We do not solve VNE directly ...
But show Hopfield‘s preprocessing capabilities

NeuroViNE‘s Hopfield Network Energy Function

Select paths
with low costs =
low energy

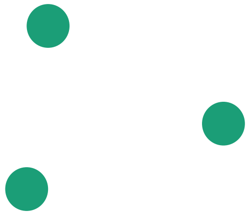
Satisfying
constraint =
low energy

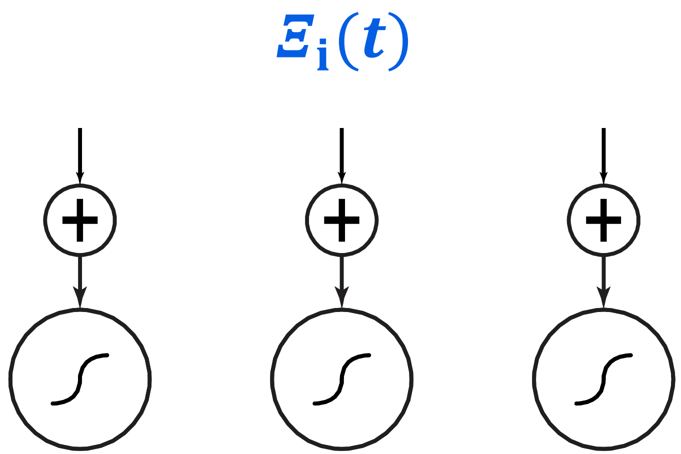
Select physical nodes
with high CPU ratio =
low energy

$$E = \boldsymbol{V}^T\left(\boldsymbol{\Psi}(\mathrm{t}) + \alpha \cdot \mathbf{T}^{\text{constraint}}\right)\boldsymbol{V} + \boldsymbol{V}^T\left(\boldsymbol{\Xi}(\mathrm{t}) + \alpha \cdot \mathbf{I}^{\text{constraint}}\right)$$

NeuroViNE‘s Hopfield Network Construction

Example for 3-Node Substrate and 2-Node Virtual Network





3 substrate nodes with CPU resource

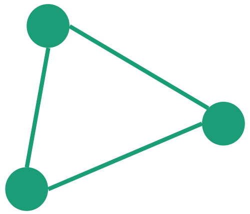
3 neurons - Input bias vector considers CPU

Node ranking

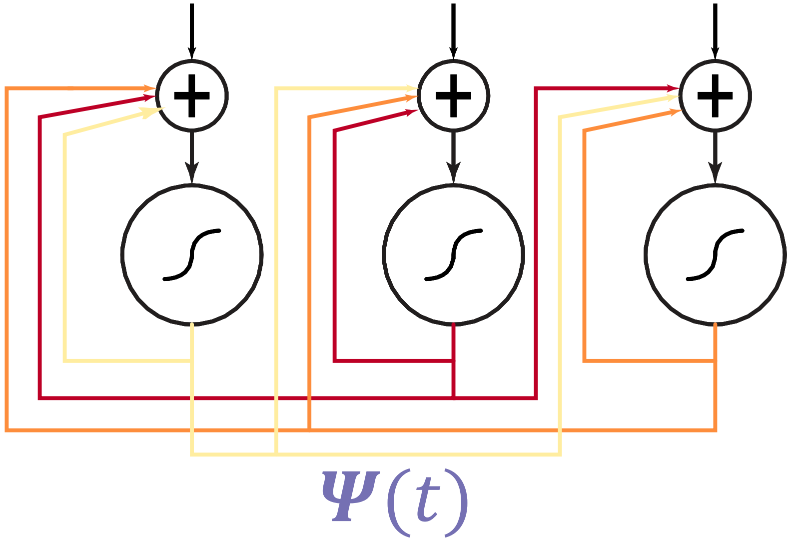
$$\Xi_i(t) = \frac{\max_{N_j \in \mathcal{N}} C_j(t) - C_i(t)}{\max_{N_j \in \mathcal{N}} C_j(t)}$$

Path Ranking

NeuroViNE's Hopfield Network Construction



3 links with datarate attributes



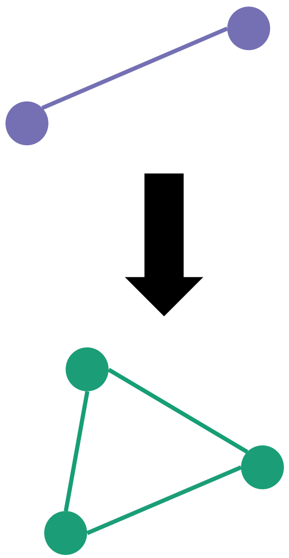
3 times 3 entries of weight matrix

Path ranking

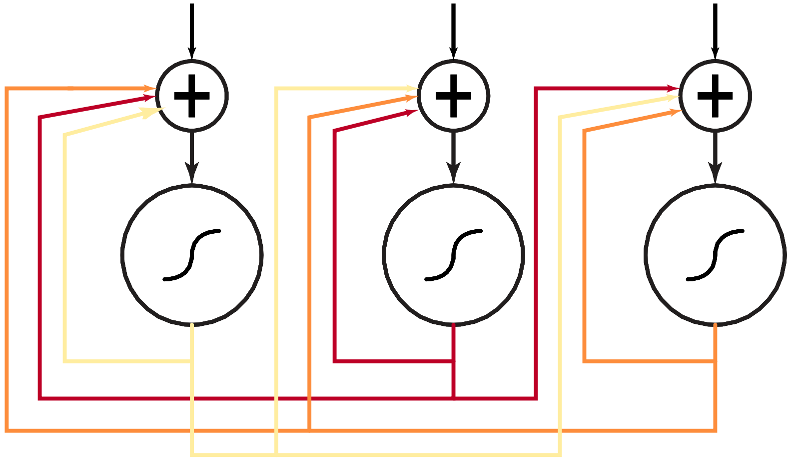
$$\Psi_{ij}(t) = \gamma \frac{D_{ij}(t)}{\max\limits_{ij} D_{ij}(t)}$$

Keeping Constraints

NeuroViNE's Hopfield Network Construction



2 Virtual nodes



2 out of 3 neurons should be chosen

Node number selection constraints

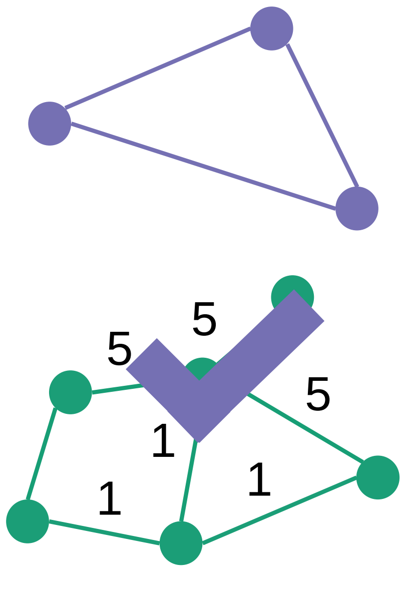
$$T_{ij}^{constraint} = \begin{cases} 1 \; if \; i \neq j \\ 0 \; otherwise \end{cases}$$

$$I_k^{constraint} = -(2 \cdot \zeta - 1)$$

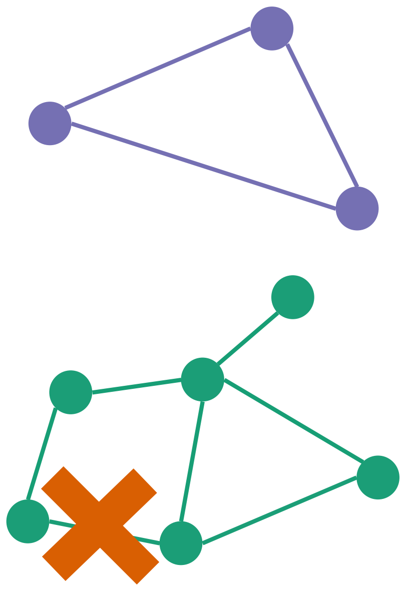
NeuroViNE‘s Hopfield Network Energy Function

$$E = \boldsymbol{V}^T\left(\boldsymbol{\Psi}(\mathrm{t}) + \alpha \cdot \mathbf{T}^{\text{constraint}}\right)\boldsymbol{V} + \boldsymbol{V}^T\left(\boldsymbol{\Xi}(\mathrm{t}) + \alpha \cdot \mathbf{I}^{\text{constraint}}\right)$$

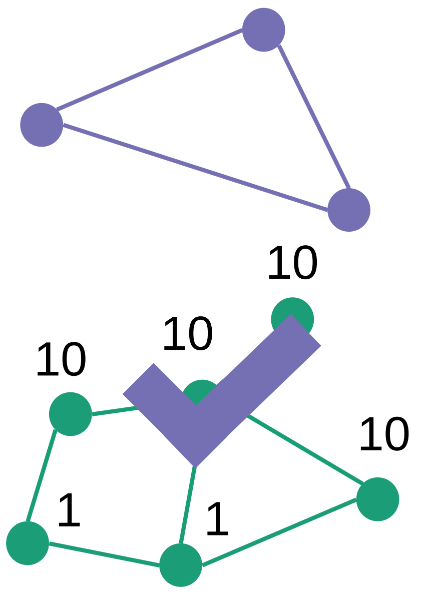
Select paths with low costs = low energy	Satisfying constraint = low energy	Select virtual nodes with high CPU ratio = low energy



Low Energy

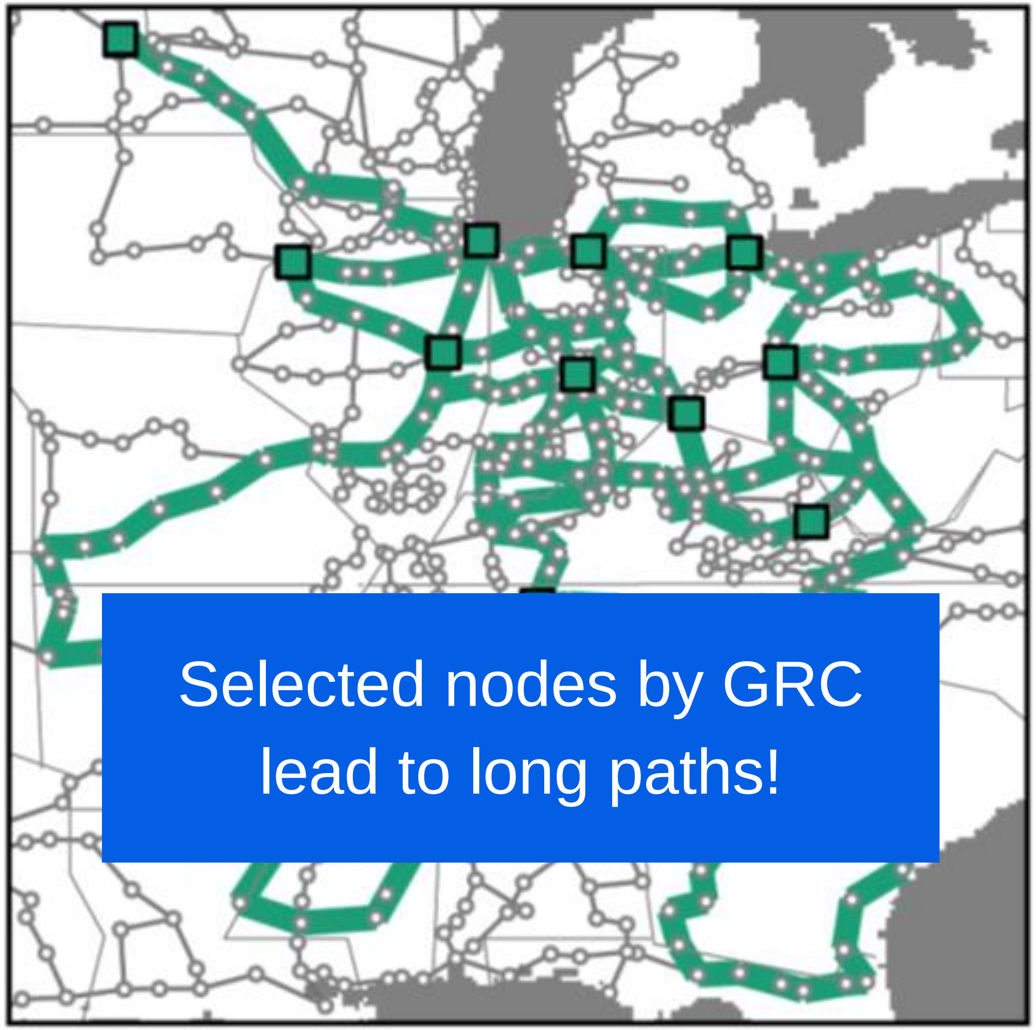


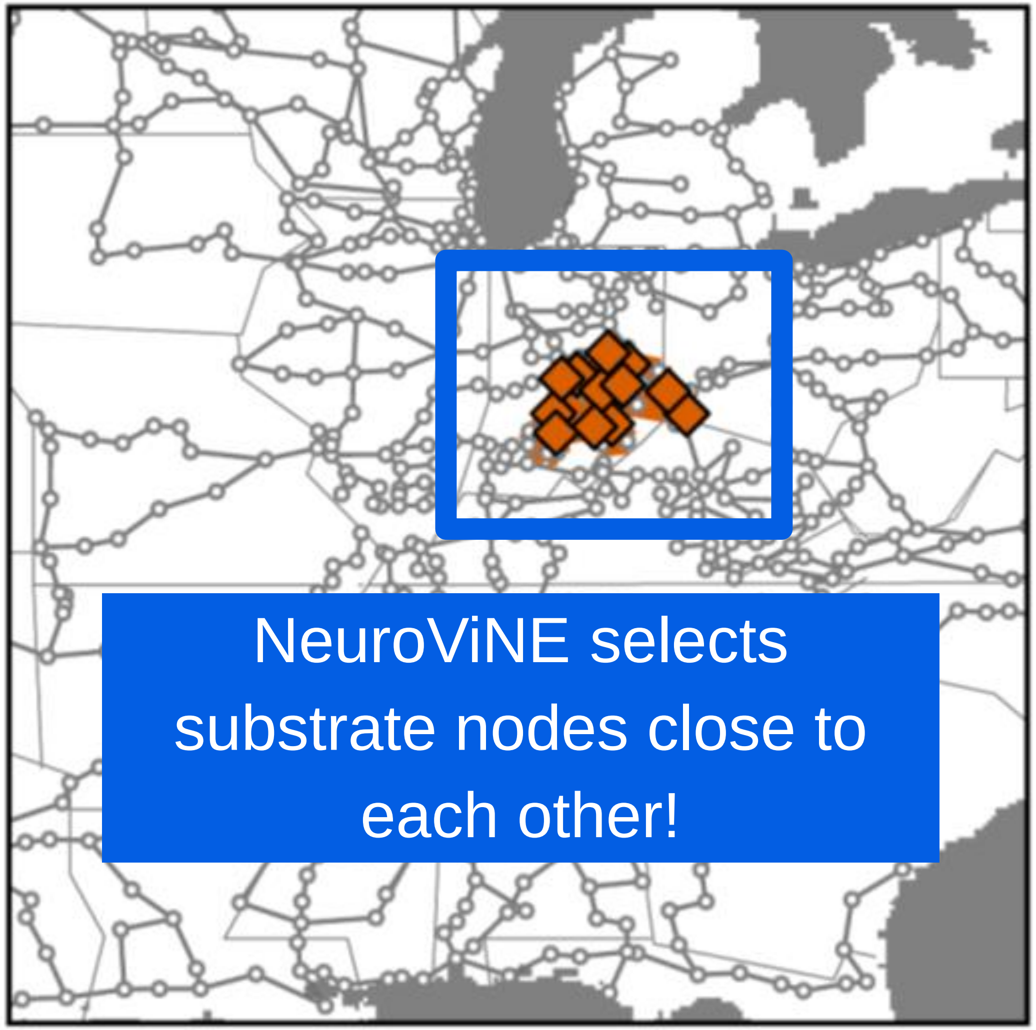
High Energy



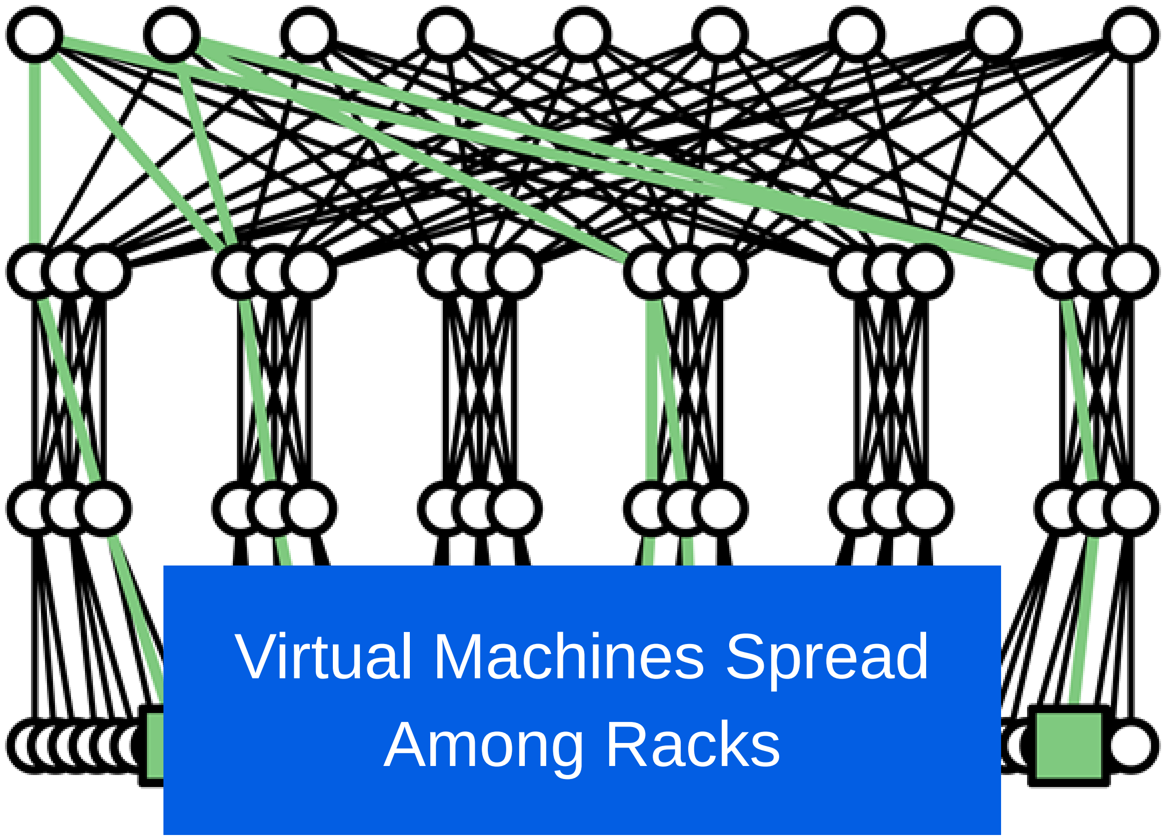
Low Energy

NeuroViNE: An Illustrative Example for GRC on 750 nodes ISP network

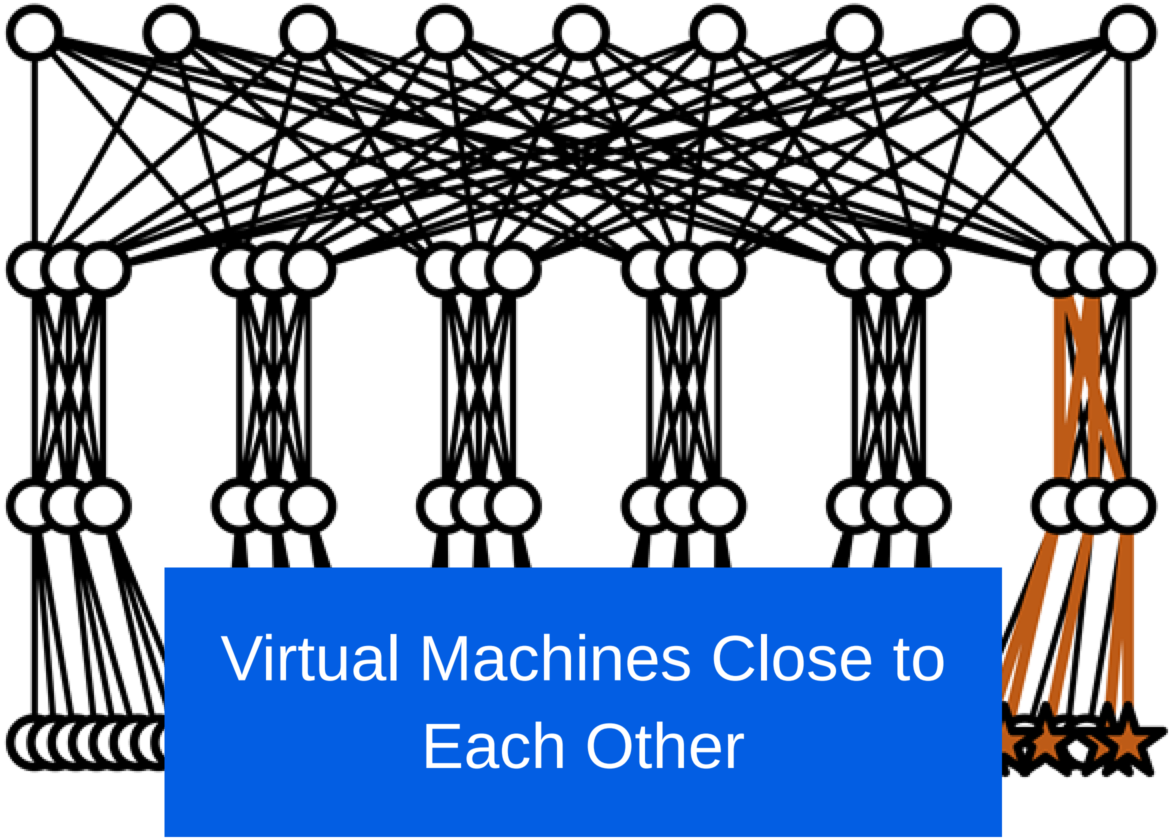




Same Behavior for Datacenters



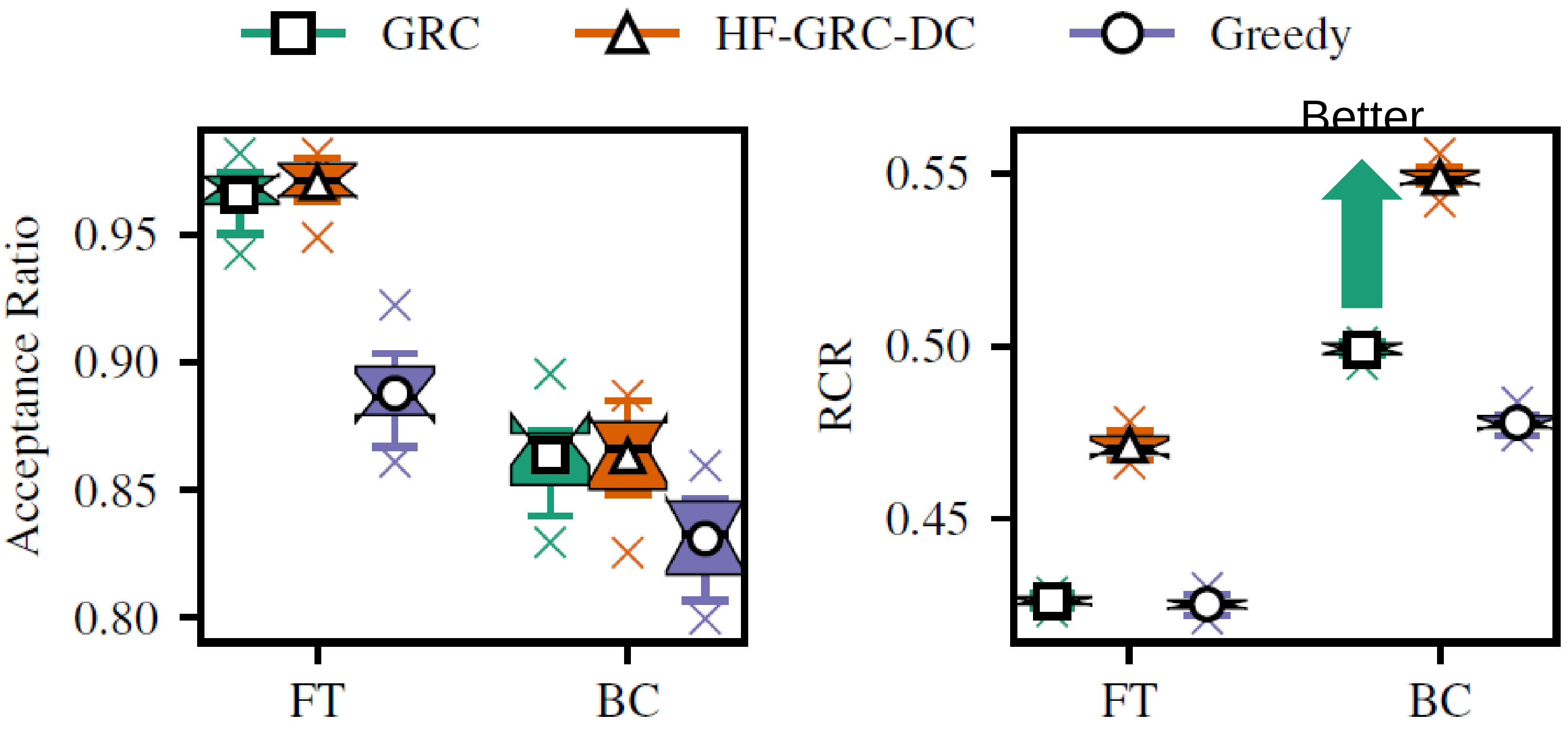
Heuristic



NeuroViNE

NeuroViNE: Efficient also in Datacenters

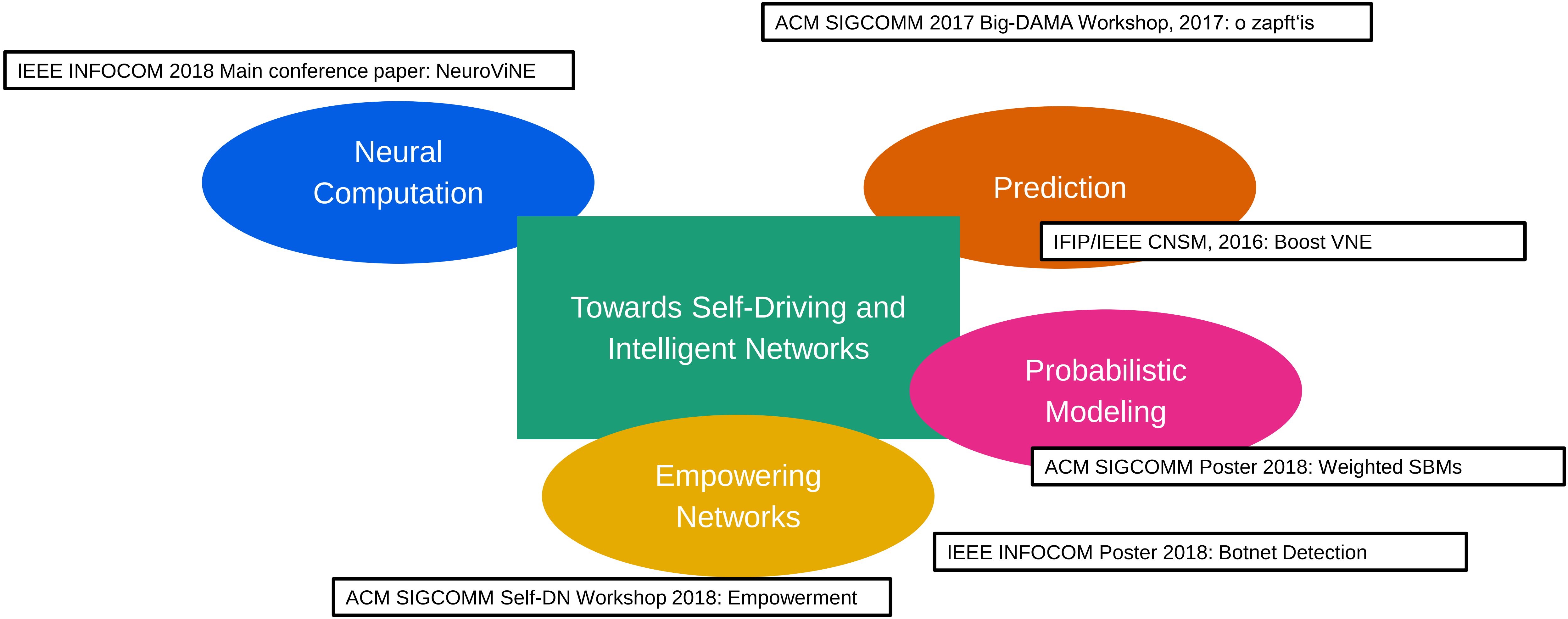
Uses a datacenter modifcation (see paper)



NeuroViNE shows similar acceptance ratios

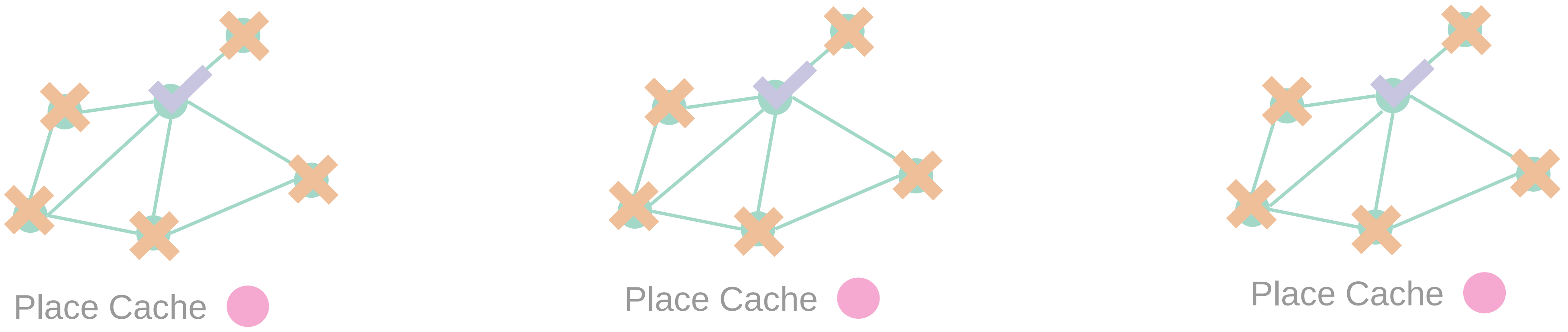
... but saves cost

Overview in this talk: o zapft‘is [7]

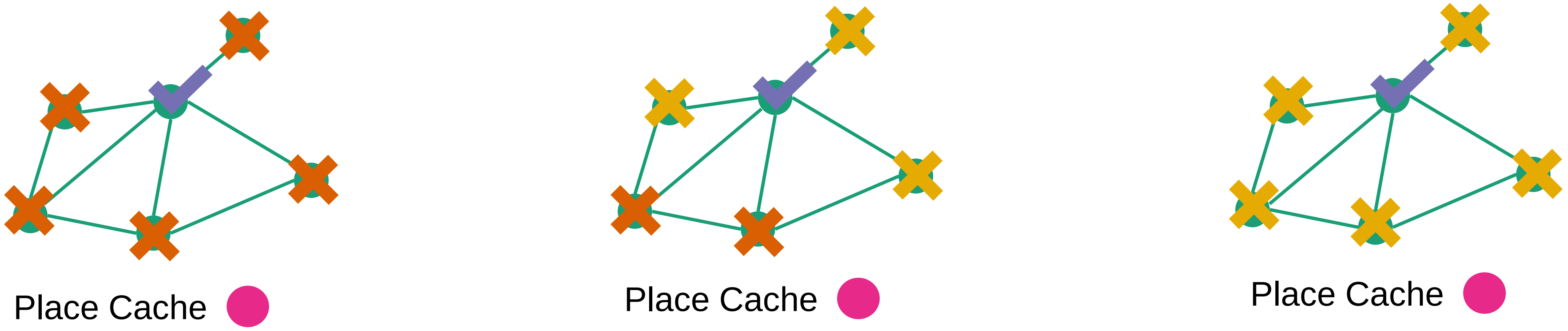


[7] A. Blenk, P. Kalmbach, S. Schmid, W. Kellerer, o'zapft is: Tap Your Network Algorithm's Big Data!, ACM SIGCOMM 2017 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks (Big-DAMA), pp. 19-24, 2017

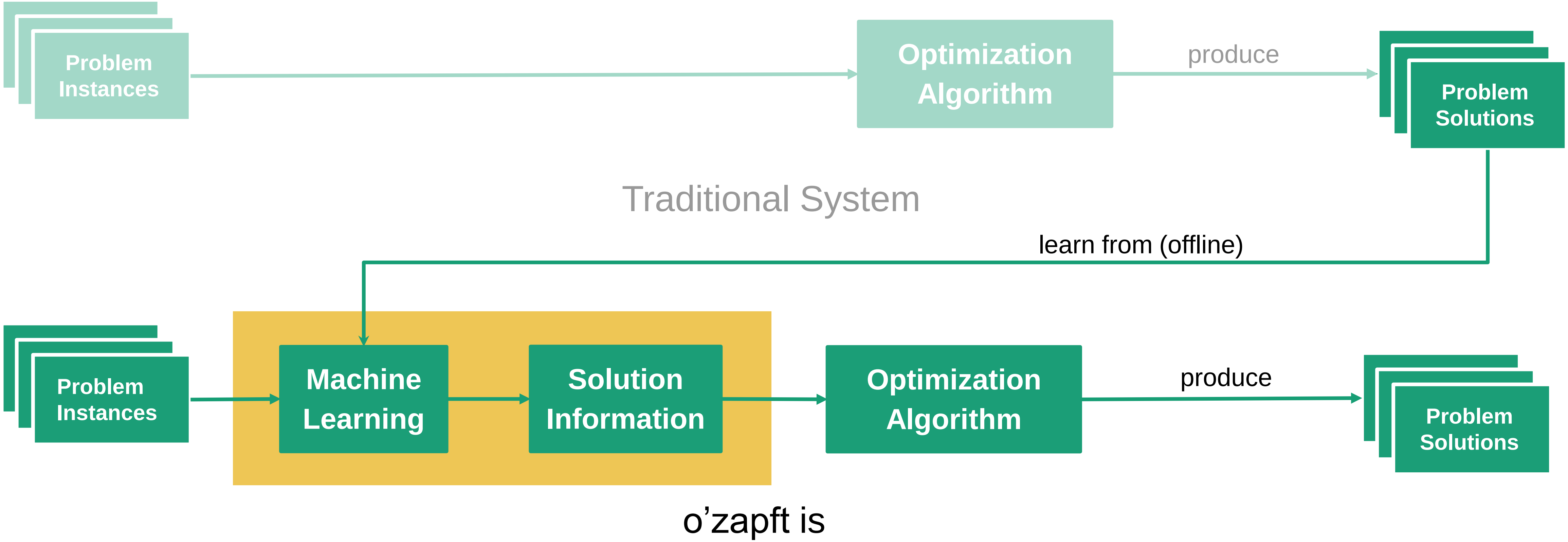
The Limitation – Fire and Forget



The Opportunity – Tap into your Algorithm’s Big Data

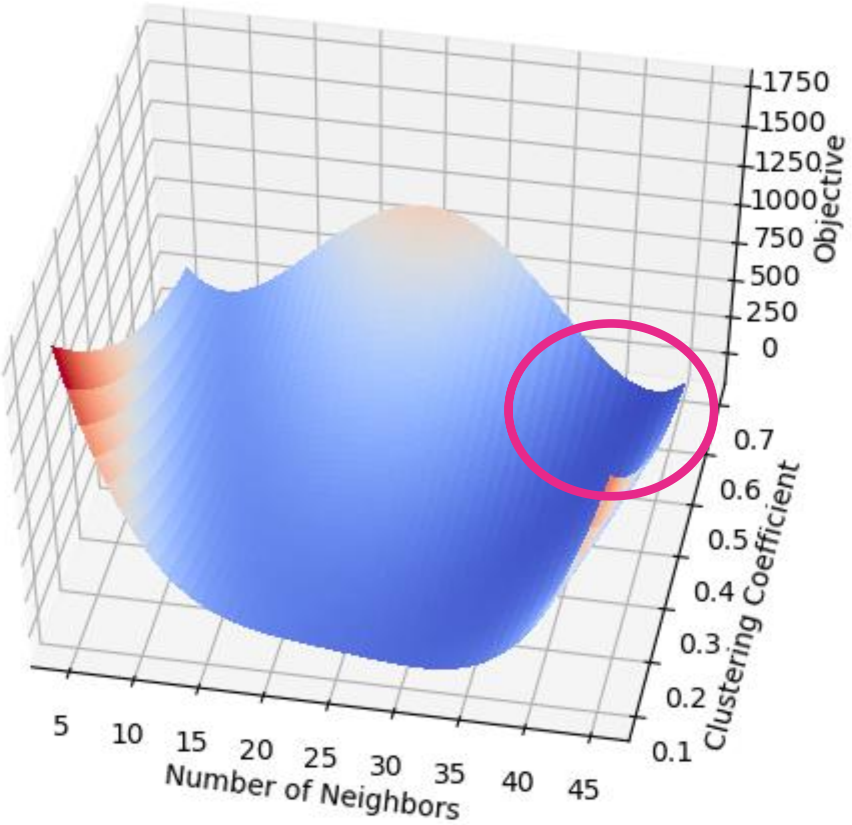


Traditional vs. Proposed System

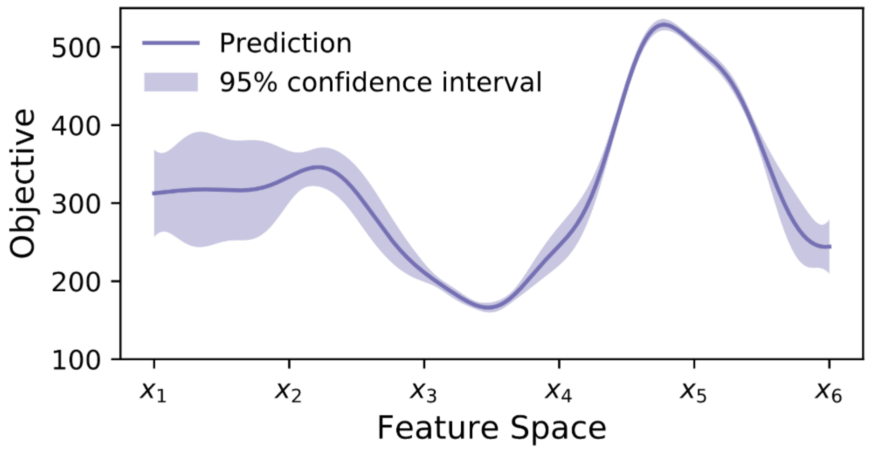


Data Available at: [8] Patrick Kalmbach, Johannes Zerwas, Michael Manhart, Andreas Blenk, Stefan Schmid, and Wolfgang Kellerer. 2017. Data on "o’zapft is: Tap Your Network Algorithm’s Big Data!". (2017). https://doi.org/10.14459/2017md1361589

Potentials

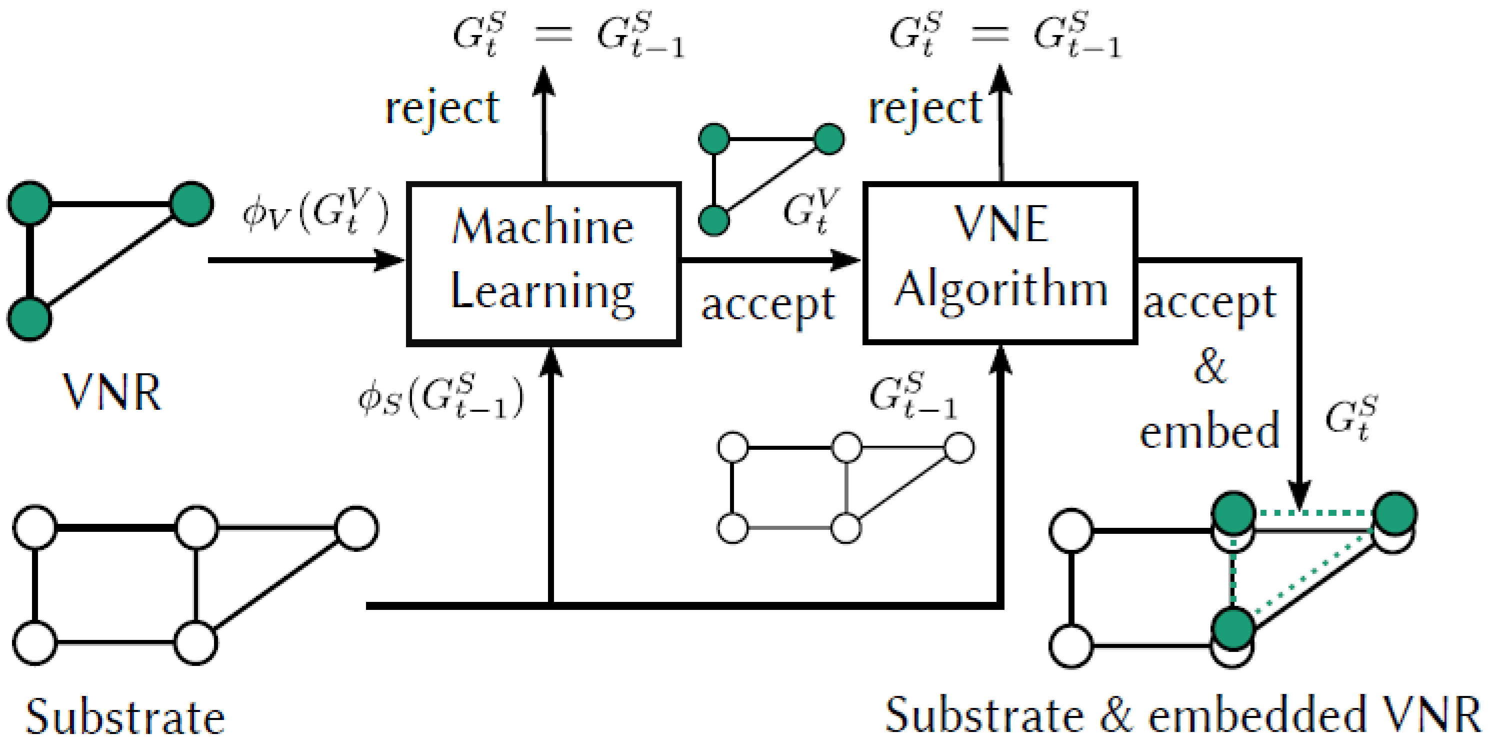


Search Space Reduction reduction/Initial Solutions



Predict Value of Objective Function

Virtual Network Embedding Prediction/Classification Pipeline [9]



- Learn and predict the acceptance and embedding cost of a VNR
- Supervised learning
- Offline training!

[9] A. Blenk, P. Kalmbach, P. van der Smagt, W. Kellerer, Boost Online Virtual Network Embedding: Using Neural Networks for Admission Control. 12th International Conference on Network and Service Management (CNSM), pp. 10-18, 2016

Learning to Accept and to Predict the Cost

Library:

- Sci-Kit Learn [9]

Graph features:

- Node degree
- Closeness
- Betweeneness
- Spectral Features

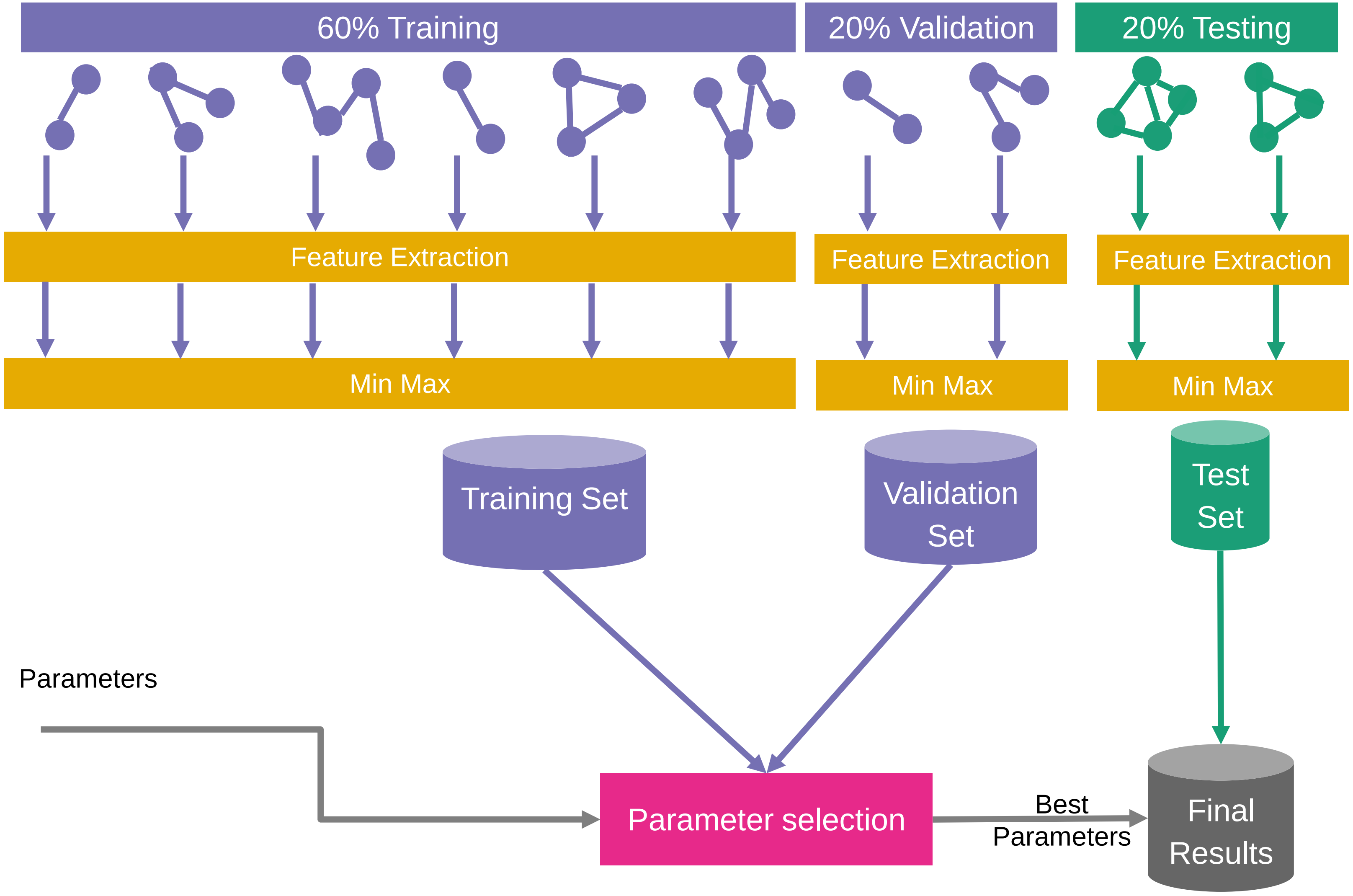
Measures:

- R^2 (goodness of fit for ML models)
- TPR/TNR/Accuracy/…

Classifier/Regressor:

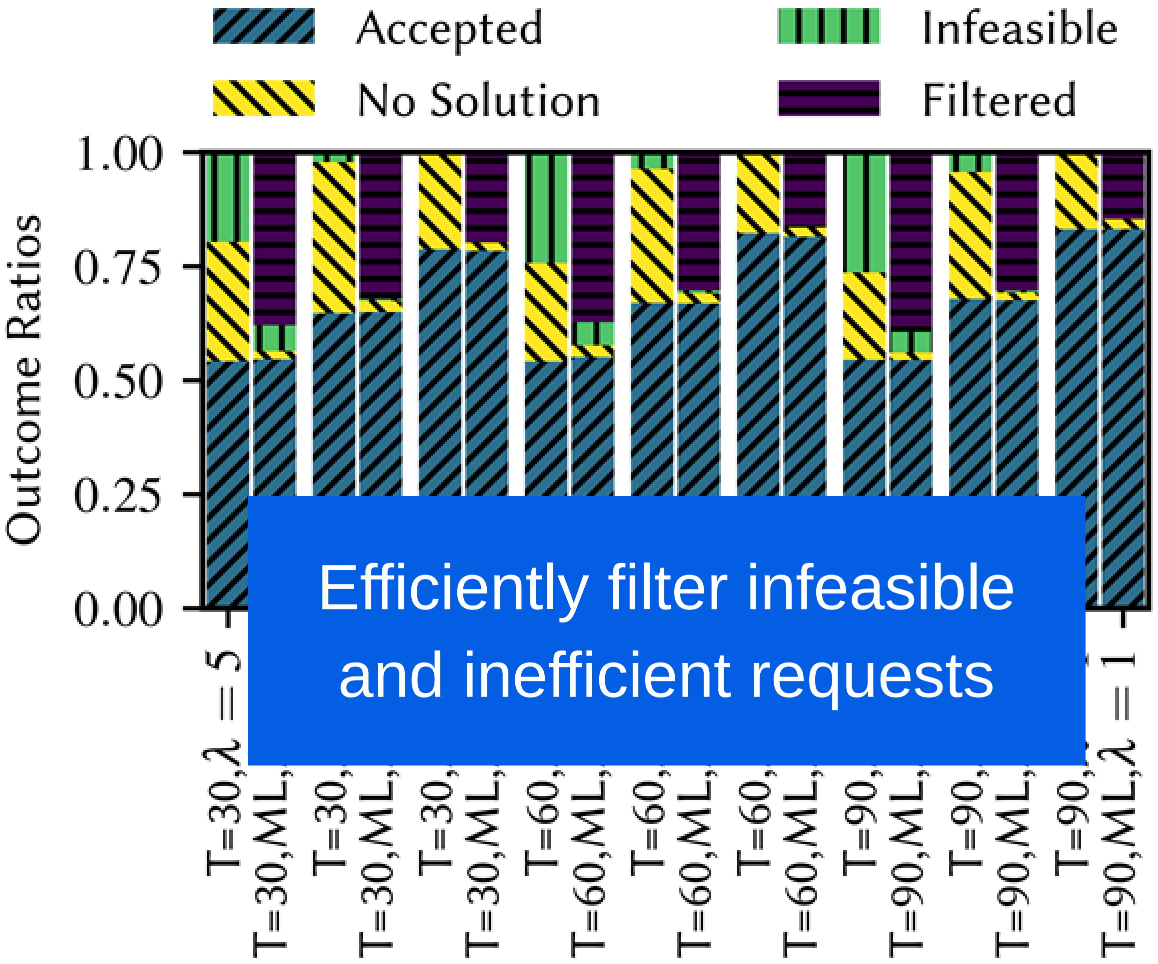
- Recurrent Neural Network (RNN)
- Linear Regression (LR)
- Bayesian Ridge Regressor (BRR)
- Random Forest Regressor (RF)
- Support Vector Regression (SVR)

Model Training and Selection:

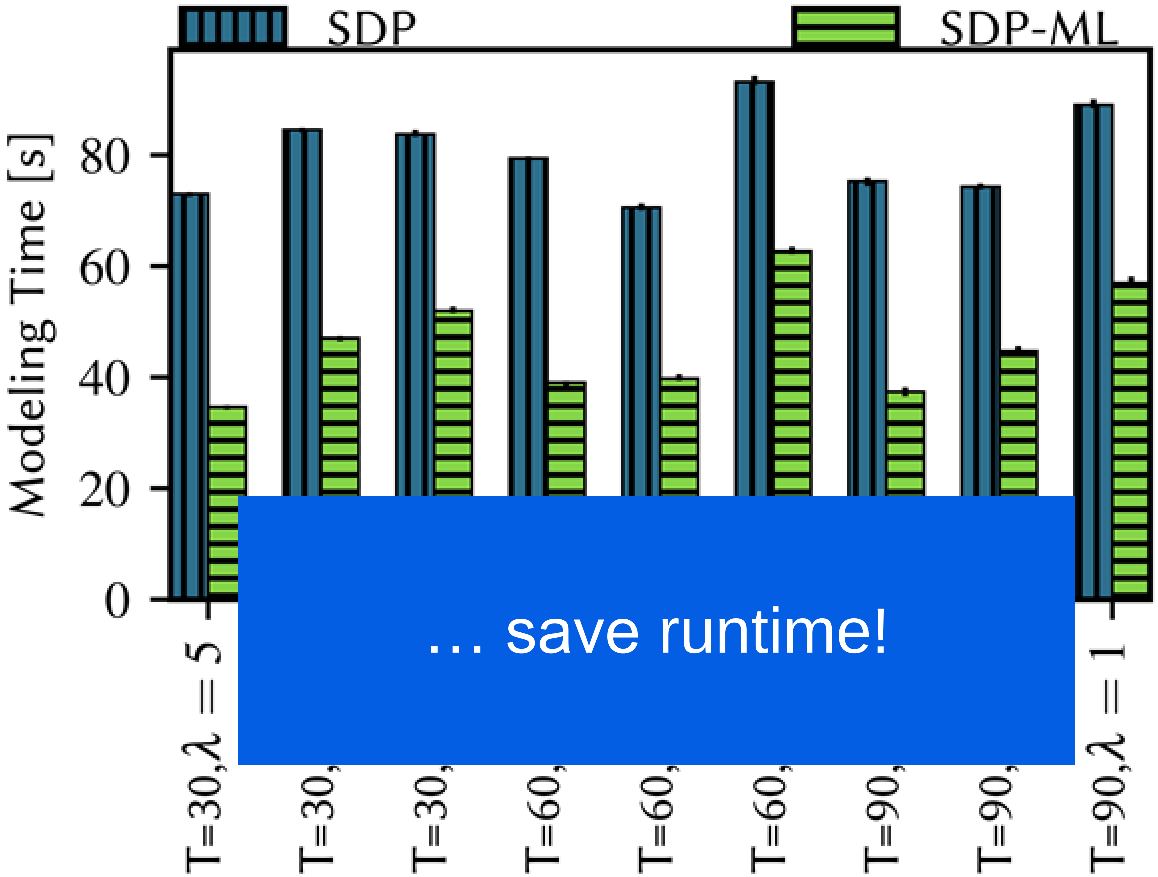


[9] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Supervised Learning & Speed-Up Results [10]



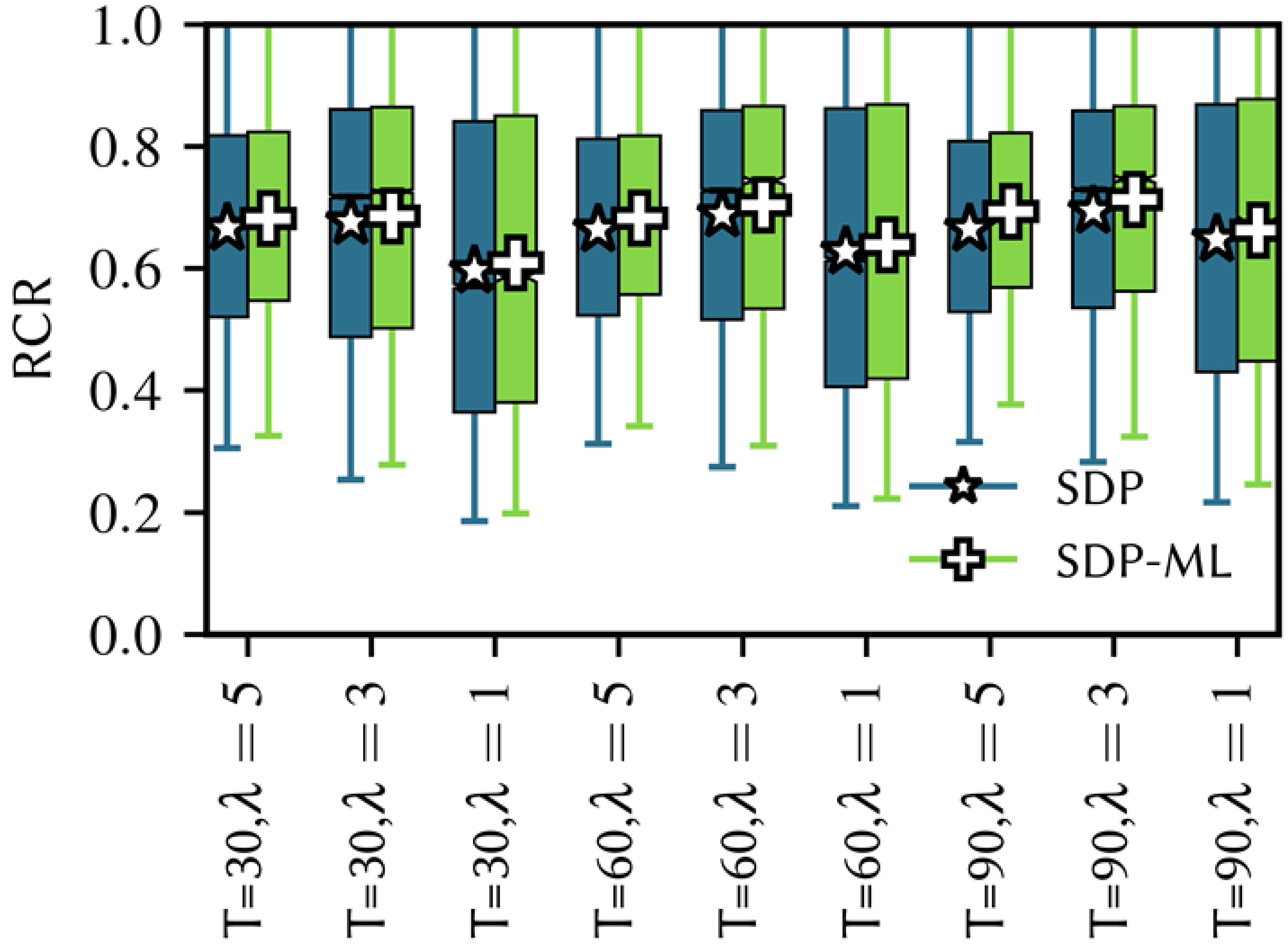
Efficiently filter infeasible and inefficient requests



… save runtime!

[10] Blenk, A. A. (2018). Towards Virtualization of Software-Defined Networks: Analysis, Modeling, and Optimization (Doctoral dissertation, Technische Universität München).

And keep the performance …



Conclusion

- Realizing flexible virtualized networks introduces new challenges
  - Overhead and interference
  - New dimensions for optimization

- This talk
  - A tool for measuring virtualization layers
  - Application of neural computation and machine learning to network algorithms

Thank you!
Questions?

Research Trailers

Where did we apply **Artificial Intelligence** (Machine Learning) so far?

Network Planning and Dimensioning

- Prepare your network for failures
- Generate network topologies with realistic characteristics

Network Monitoring

- Create network service graphs
- Detect anomalies

Network Resource Allocation Algorithms

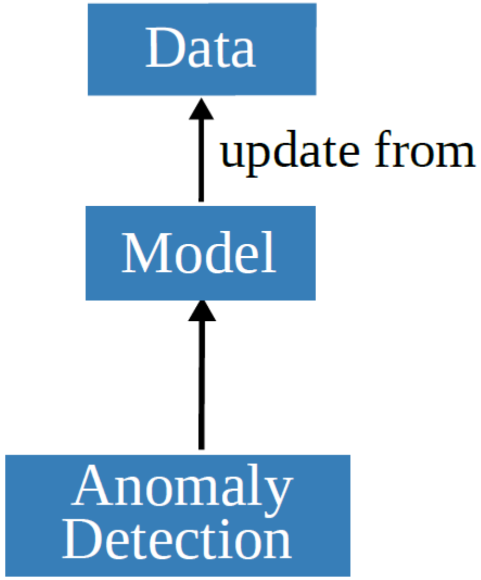
- Virtual network provisioning
- Function placement
- Admission control
- Flow routing

Research Towards Self-Driving Networks

Network Monitoring

- (Weighted) Stochastic Block Models
- Machine Learning for training (unsupervised)

- Who communicates with whom
  → Create service graph layout
- Plan and benchmarking
  → Generate realistic communication patterns
- Communication pattern changes over time
  → Detect anomalies (abnormal bot communication)

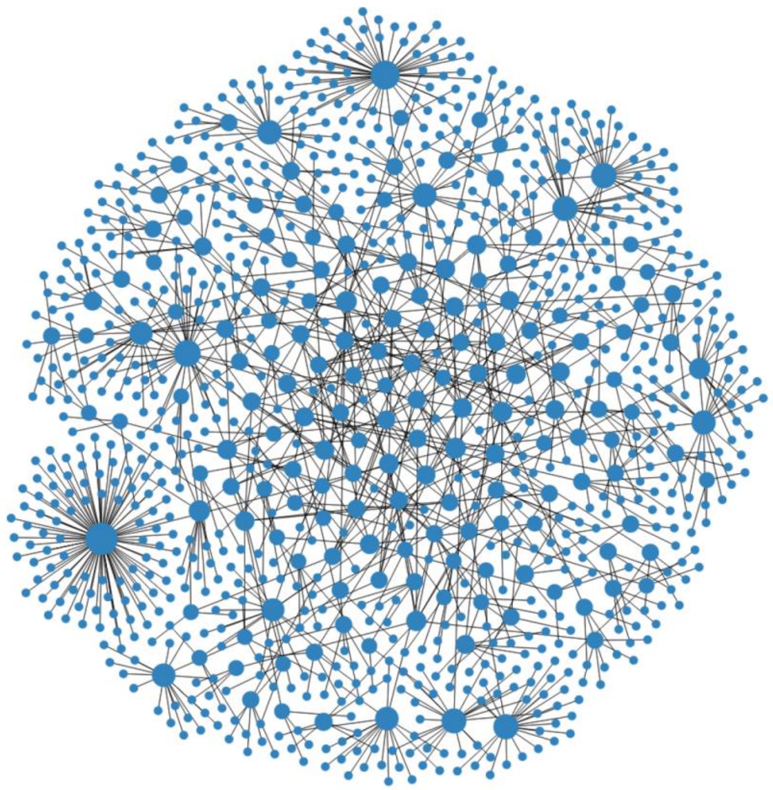


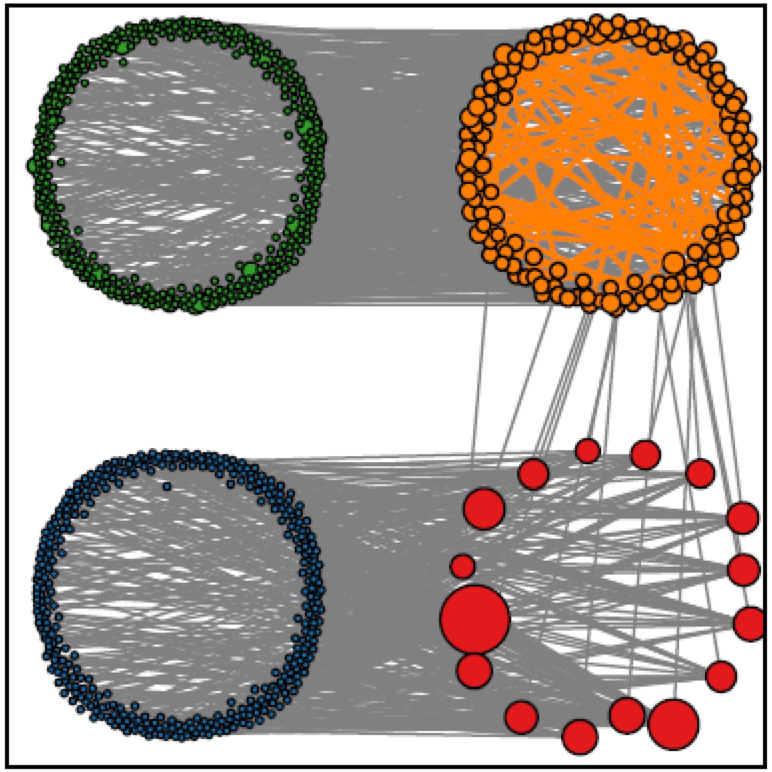
SBM has many applications!

IEEE AnNet Workshop 2017: Generating Topologies with SBMs [11]	IEEE INFOCOM Poster 2018: Botnet Detection [12]	ACM SIGCOMM Poster 2018: Weighted SBMs [13]

[11] P. Kalmbach, A. Blenk, M. Kluegel, W. Kellerer, Generating Synthetic Internet- and IP-Topologies using the Stochastic-Block-Model. 2nd IFIP/IEEE International Workshop on Analytics for Network and Service Management (AnNet), 2017
[12] P. Kalmbach, A. Blenk, S. Schmid, W. Kellerer, Themis: Data Driven Approach to Botnet Detection. 37th IEEE Conference on Computer Communications (INFOCOM), 2018
[13] P. Kalmbach, L. Gleiter, J. Zerwas, A. Blenk, W. Kellerer, Modeling IP-to-IP communication using the Weighted Stochastic Block Model. ACM SIGCOMM 2018 Conference Posters and Demos, 2018

SBM in Action

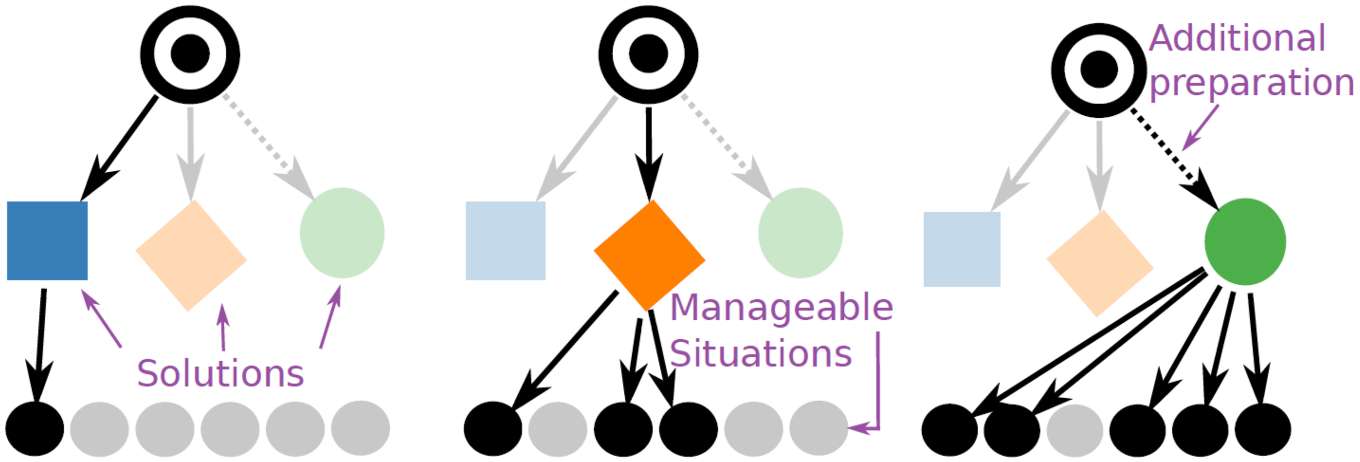




- Router Level Graph of the Internet
- 939 nodes, 988 edges

- Intelligent grouping based on communication patterns only

- Empower your network to be prepared
  - For failures
  - For changing traffic
  - For new services
  - ...



(a) Simple. (b) Robust. (c) Empowerment.

ACM SIGCOMM Self-DN Workshop 2018: Empowerment [14]

Empowerment towards Network Intelligence?

[14] P. Kalmbach, J. Zerwas, P. Babarczi, A. Blenk, W. Kellerer, S. Schmid, Empowering Self-Driving Networks. Proceedings of the Afternoon Workshop on Self-Driving Networks - SelfDN 2018, ACM Press, 2018