APPLICATIONS OF REINFORCEMENT LEARNING TO ROUTING AND VIRTUALIZATION IN COMPUTER NETWORKS

Soroush Haeri

Communication Networks Laboratory
http://www.ensc.sfu.ca/~ljilja/cnl/
Simon Fraser University
Vancouver, British Columbia, Canada

Roadmap

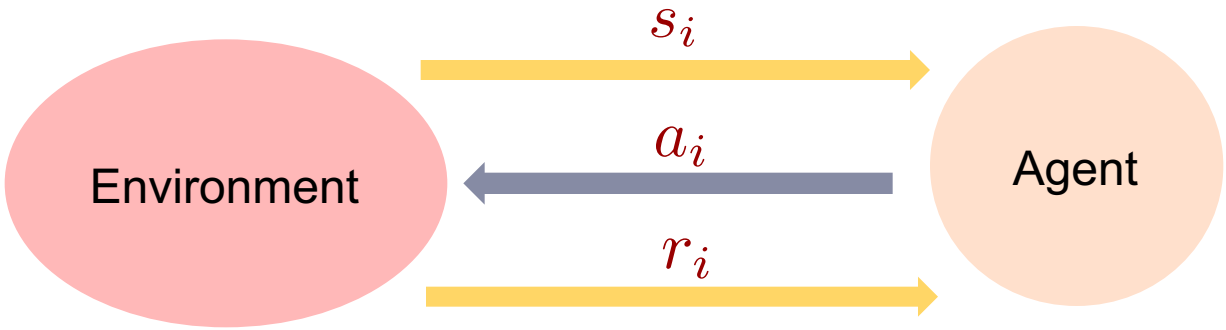
- Introduction
- Deflection routing
- Virtual network embedding
- Conclusion
- Publications and references

Introduction: Machine Learning

- Machine learning:
  - improves agents' behaviors by analyzing examples of desirable interactions
- Three main learning categories:
  - supervised
  - unsupervised
  - reinforcement

Introduction: Reinforcement Learning (RL)

- RL agent observes the state of the environment s_i
- It then selects an appropriate action a_i
- The environment generates a reinforcement signal r_i and transmits it to the agent
- The agent employs the reinforcement signal to improve its subsequent decisions



Introduction: RL and Computer Networks

- Not widely used in conventional computer networks:
  - RL algorithms were mostly designed to solve problems with a high degree of complexity
  - early computer networks were mostly operated in controlled environments: low degree of complexity
  - RL algorithms required high computational power and/or large memory
  - network devices had rather limited computational power and memory

Introduction: Recent Trends

- Recent trend in computer networking:
  - Software-Defined Networks
  - centralized implementation of network control logic
- Consequences:
  - more powerful network control units
  - new applications and challenges

Deflection Routing vs. VNE

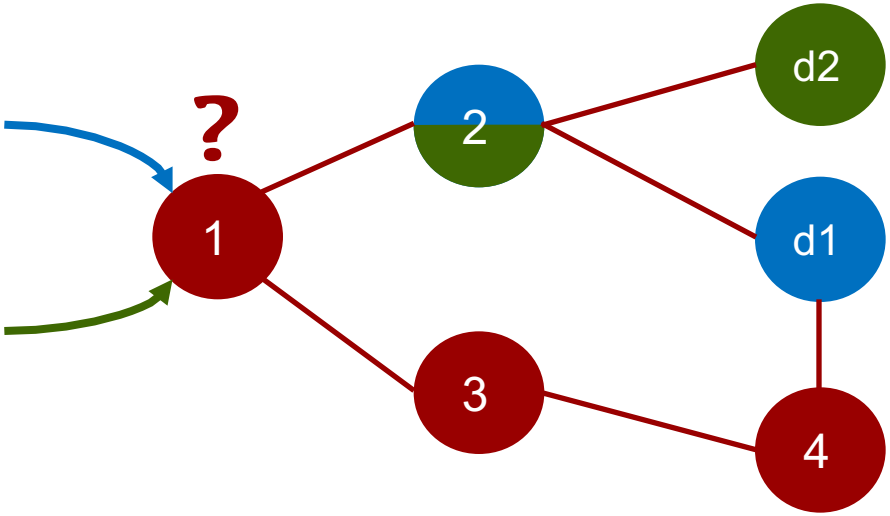
- Deflection routing:
  - classical networking question
  - decision making instances may be intermittent (no cause and effect relationship), hence:
    - more challenging learning problem
- VNE:
  - recent networking challenge
  - classical learning problem

Roadmap

- Introduction
- Deflection routing
- Virtual network embedding
- Conclusion
- Publications and references

Optical Burst Switching (OBS)

- Enables an all-optical switching
  - eliminates optical/electrical/optical conversions
- Contention is one of the main challenges (no optical buffers)



Deflection Routing

- A contending flow is routed through the optimal link defined by the routing table
- Temporarily deflect the other flow away from the path that is prescribed by the routing table
- Deflection routing algorithms coexist in the network along with underlying routing protocols

C. Qiao and M. Yoo, "Optical burst switching (OBS)–a new paradigm for an optical Internet," *J. of High Speed Netw.*, vol. 8, no. 1, pp. 69–84, Mar. 1999.

A. S. Acampora and S. I. A. Shah, "Multihop lightwave networks: a comparison of store-and-forward and hot-potato routing," in *Proc. IEEE INFOCOM*, vol. 1, Bal Harbour, FL, USA, Apr. 1991, pp. 10–19.

Deflection Routing by Reinforcement Learning: iDef Framework

- Facilitate development of reinforcement learning-based deflection routing protocols
- Two main components:
  - signaling algorithm
  - learning algorithm
- Implemented in ns-3 and made publicly available

(2016, Jan.) iDef ns-3 implementation repository. [Online]. Available: http: //bitbucket.org/shaeri/hmm-deflection/.

Existing Reinforcement Learning-Based Deflection Routing Algorithms

- Q-learning Path Selection algorithm
- Reinforcement Learning-Based Deflection Routing Scheme (RLDRS)
- Uses Q-learning
- One entry for every destination in the network
  - complexity: size of the network

Y. Kiran, T. Venkatesh, and C. Murthy, "A reinforcement learning framework for path selection and wavelength selection in optical burst switched networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 18–26, Dec. 2007.

A. Belbekkouche, A. Hafid, and M. Gendreau, "Novel reinforcement learning-based approaches to reduce loss probability in buffer-less OBS networks," *Comput. Netw.*, vol. 53, no. 12, pp. 2091–2105, Aug. 2009.

Q-learning: Deficiencies

- Table-Based (Q-table) learning algorithm
- One table entry for every state-action pair
- Deficiencies:
  - no path recovery and reselection
  - does not use reinforcement signals efficiently

C. J. C. H. Watkins and P. Dayan, "Technical note, Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

S. P. M. Choi and D. Y. Yeung, "Predictive q-routing: a memory-based reinforcement learning approach to adaptive traffic control," in *Advances in Neural Inform. Process. Syst.*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA, USA: The MIT Press, 1996, vol. 8, pp. 945–951.

Contributions: Algorithms

- Proposed the Predictive Q-learning Deflection Routing (PQDR) algorithm
  - complexity: size of the network
  - uses predictive Q-learning:
    - enables path recovery and reselection
- Proposed the Node Degree Dependent (NDD) signaling algorithm
  - complexity: node degree

S. Haeri, M. Arianezhad, and Lj. Trajković, "A predictive q-learning-based algorithm for deflection routing in buffer-less networks," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, Manchester, UK, Oct. 2013, pp. 764–769.

S. Haeri, W. W.-K. Thong, G. Chen, and Lj. Trajković, "A reinforcement learning- based algorithm for deflection routing in optical burst-switched networks," in *Proc. The 14th IEEE Int. Conf. Inform. Reuse and Integration (IRI 2013)*, San Francisco, USA, Aug. 2013, pp. 474–481.

Contributions

- Proposed a Neural Network (NN) and an Episodic Neural Network (ENN) learning algorithms
  - use the reinforcement signals more efficiently

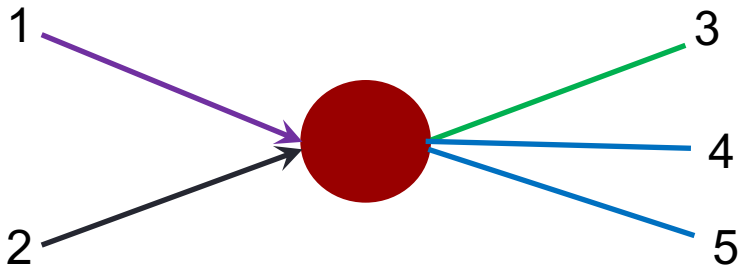
S. Haeri and Lj. Trajković, "Intelligent deflection routing in buffer-less networks," *IEEE Tran. Cybern.*, vol. 45, no. 2, pp. 316–327, Feb. 2015.

The Node Degree Dependent Algorithm

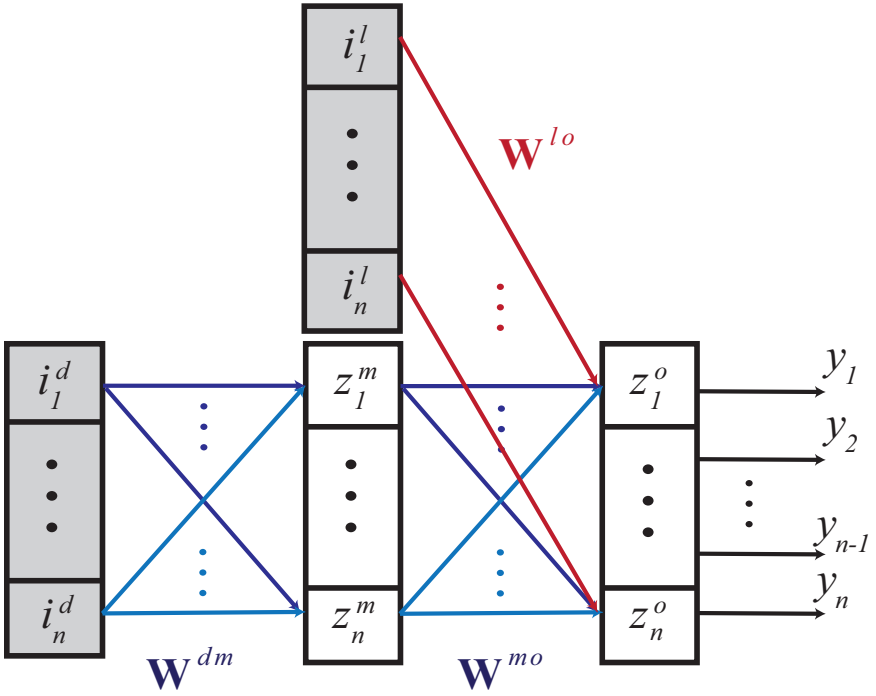
- Defines the state of the system locally at each node
- Consider a buffer-less node with *n* outgoing links
- Two binary vectors:
  - $\mathbf{I}^l = [i_1^l \ \ldots \ i_n^l]$: if the k^{th} outgoing link of the node is blocked, then $i_k^l = 1$
    - blocking state of a node's outgoing links
  - $\mathbf{I}^d = [i_1^d \ \ldots \ i_n^d]$: if the the burst that is to be deflected contends for the j^{th} outgoing link, then $i_j^d = 1$

State Definition: Example

- A buffer-less node with 5 links
- Two incoming flows from violet and black links contending for the green link
- The blue links are idle
- $\mathbf{I}^l = [1\ 1\ 1\ 0\ 0]$
- $\mathbf{I}^d = [0\ 0\ 1\ 0\ 0]$



Neural Networks for Deflection Routing



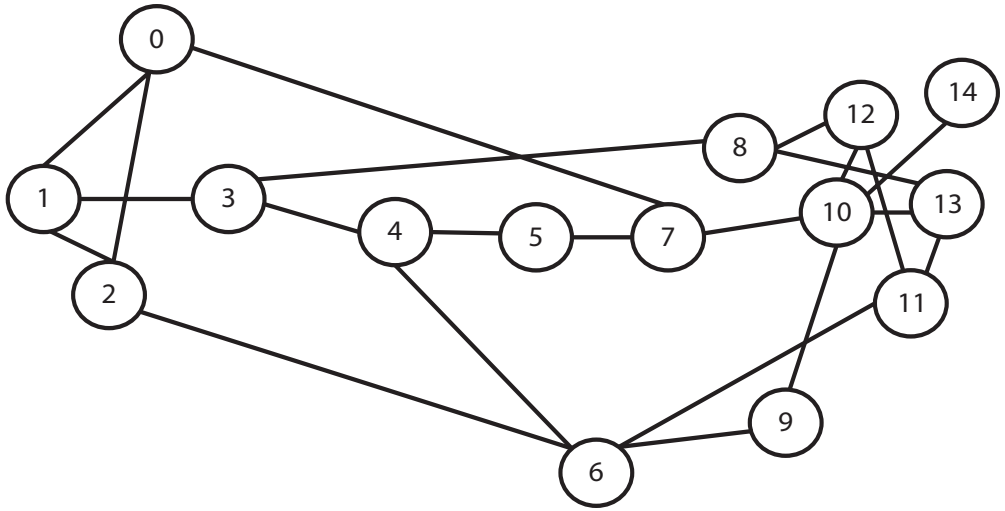
$$\mathbf{W}^{dm} = \begin{bmatrix} w_{11}^{dm} & w_{12}^{dm} & \cdots & w_{1n}^{dm} \\ w_{21}^{dm} & w_{22}^{dm} & \cdots & w_{2n}^{dm} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{dm} & w_{n2}^{dm} & \cdots & w_{nn}^{dm} \end{bmatrix}$$

$$\mathbf{W}^{mo} = \begin{bmatrix} w_{11}^{mo} & w_{12}^{mo} & \cdots & w_{1n}^{mo} \\ w_{21}^{mo} & w_{22}^{mo} & \cdots & w_{2n}^{mo} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{mo} & w_{n2}^{mo} & \cdots & w_{nn}^{mo} \end{bmatrix}$$

$$\mathbf{W}^{lo} = \begin{bmatrix} w_{11}^{lo} & 0 & \cdots & 0 \\ 0 & w_{22}^{lo} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & w_{nn}^{lo} \end{bmatrix}$$

Simulations

- National Science Foundation (NSF) network topology (64 wavelengths):
  - burst loss probability
  - average number of hops



NSF network after the 1989 transition

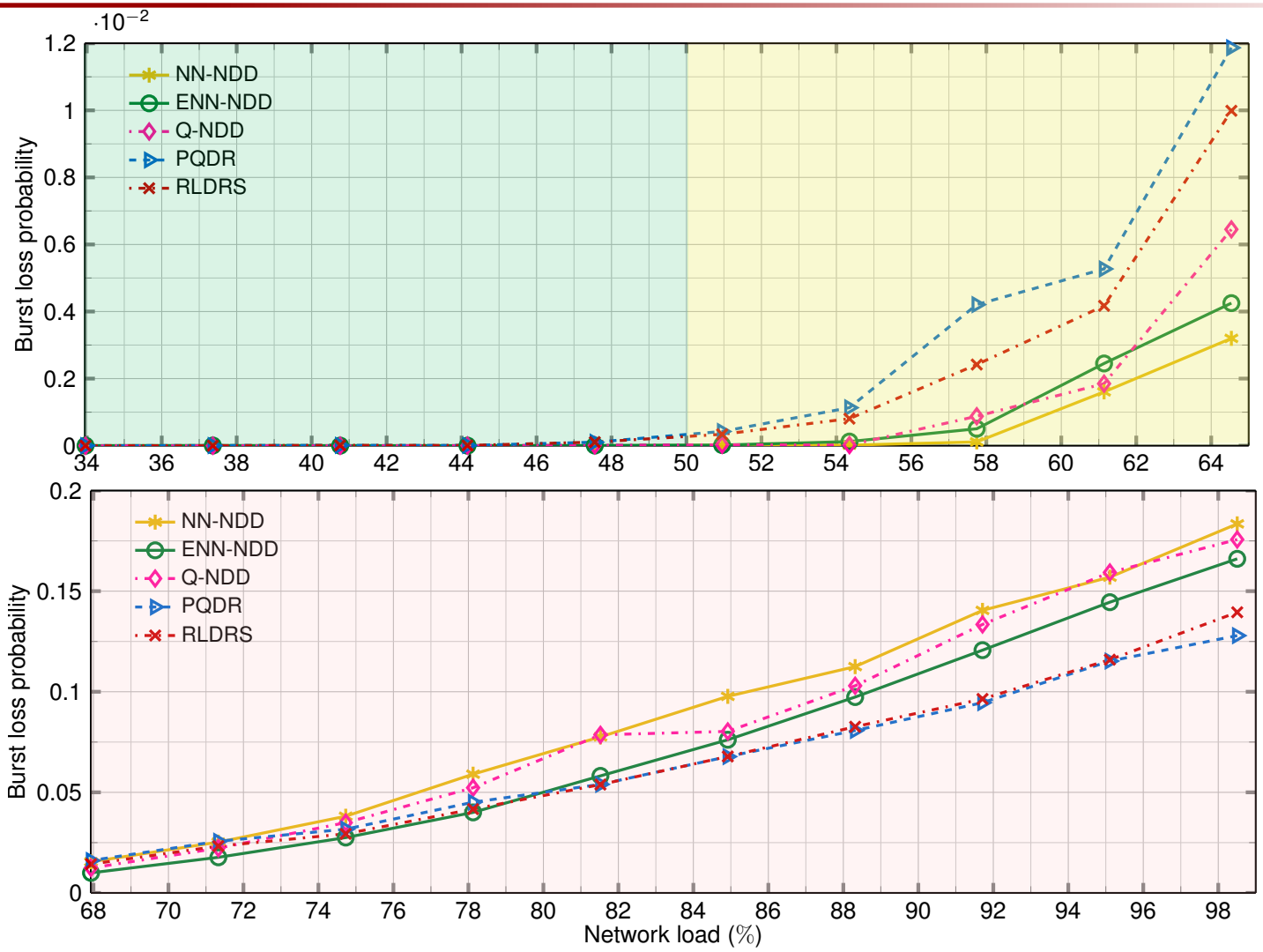
Results: Burst loss Probability (Prequel)

- Internet backbone was engineered to keep link load levels < 50%
- Studies show that the overloads of > 50% occur < 0.2% of a link life-time
- Goal:
  - achieve good performance in low to moderate traffic loads

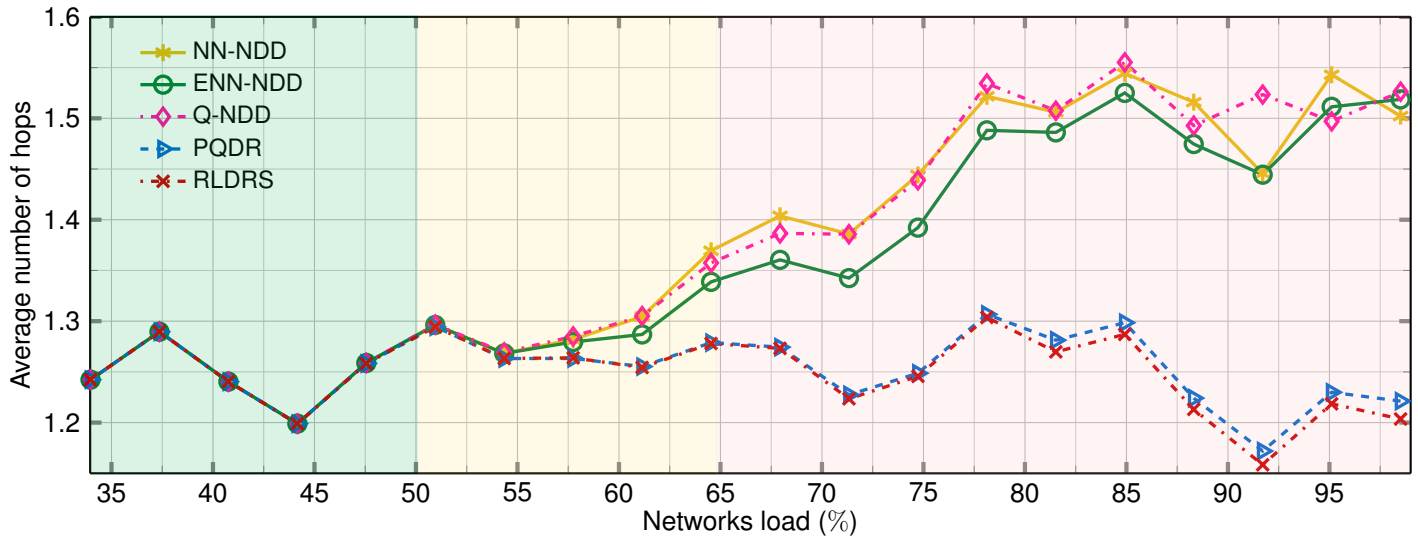
S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE INFOCOM*, vol. 1, Stanford, CA, USA, Mar. 2003, pp. 406–416.

C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Netw.*, vol. 17, no. 6, pp. 6–16, Dec. 2003.

Results: Burst Loss Probability



Results: Average Number of Hops



- The RLDRS and PQDR signaling algorithms take into account the number of hops to destination:
  - smaller average end-to-end delay and average number of hops traveled by bursts

Roadmap

- Introduction
- Deflection routing
- Virtual network embedding
- Conclusion
- Publications and references

Network Virtualization

- Enables coexistence of multiple virtual networks on a physical infrastructure
- Virtualized network model divides the role of the Internet Service Providers (ISPs) into:
  - Infrastructure Providers (InPs):
    - manage the physical infrastructure
  - Service Providers (SPs):
    - aggregate resources from multiple InP into multiple Virtual Networks (VNs)

Virtual Network Embedding

- Virtual Network Embedding (VNE) allocates substrate network resources to VNs
- InPs' revenues depend on VNE efficiency
- VNE problem may be reduced to the multi-way separator problem:
  - NP-hard problem
  - optimal solutions are only feasible for small problems

M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *Comput. Commun. Rev.*, vol. 38, no. 2, pp. 19–29, Mar. 2008.

VNE Solution

- Two subproblems:
  - Virtual Node Mapping (VNoM): maps virtual nodes to substrate nodes
  - Virtual Link Mapping (VLiM): maps virtual links to substrate paths
- VNE algorithms address the VNoM while solving the VLiM using algorithms:
  - Shortest-Path (SP)
  - Multicommodity Flow (MCF)

VNE Formulation: Constrains

- Substrate network graph: $G^s(N^s, E^s)$
- Resources:
  - substrate nodes: CPU capacity $\mathcal{C}(n^s)$
  - substrate links: bandwidth $\mathcal{B}(e^s)$
- Virtual network graph: $G^{\Psi_i}(N^{\Psi_i}, E^{\Psi_i})$
- Resource requirements:
  - virtual nodes: CPU capacity $\mathcal{C}(n^{\Psi_i})$
  - virtual links: bandwidth $\mathcal{B}(e^{\Psi_i})$

VNE Objective

- We consider embedding one VNR at a time
- Maximize the profit of InPs
- Contributing factors to the generated profit:
  - embedding revenue
  - embedding cost
  - acceptance ratio

M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.

VNE Objective: Revenue

- Maximize revenue:

$$\mathbf{R}(G^{\Psi_i}) = w_c \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + w_b \sum_{e^{\Psi_i} \in E^{\Psi_i}} \mathcal{B}(e^{\Psi_i})$$

  - w_c: weights for CPU requirements
  - w_b: weight for bandwidth requirements
  - Assumption: $w_c = w_b = 1$
- Generated revenue is not a function of the embedding configuration

X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *Comput. Commun. Rev.*, vol. 41, pp. 38–47, Apr. 2011.

VNE Objective: Cost

- Minimize the cost:

$$\mathbf{C}(G^{\Psi_i}) = \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + \sum_{e^{\Psi_i} \in E^{\Psi_i}} \sum_{e^s \in E^s} f_{e^s}^{e^{\Psi_i}}$$

  - $f_{e^s}^{e^{\Psi_i}}$: total bandwidth of the substrate link e^s allocated to virtual link e^{Ψ_i}
- Cost depends on the embedding configuration

VNE Objective: Acceptance Ratio

- Maximize acceptance ratio:

$$p_a^\tau = \frac{|\Psi^a(\tau)|}{|\Psi(\tau)|}$$

  - $|\Psi^a(\tau)|$: number of accepted Virtual Network Requests (VNRs) in a given time interval τ
  - $|\Psi(\tau)|$: number of all arrived VNRs in τ

VNE Objective Function

- We propose to maximize:

$$\mathcal{F}(\Psi_i) = \begin{cases} \mathbf{R}(G^{\Psi_i}) - \mathbf{C}(G^{\Psi_i}) & \text{successful embeddings} \\ \Gamma & \text{otherwise} \end{cases}$$

  - Γ : large negative penalty for unsuccessful embedding
- Proved the upper bound:

$$\mathcal{F}(\Psi_i) \leq 0$$

Available VNE Algorithms: Vine and Global Resource Capacity (GRC)

- Vine (R-Vine and D-Vine):
  - formulate VNE problem as a Mixed Integer Program (MIP)
  - use Multicommodity Flow algorithm for solving VLiM
- GRC:
  - Node-ranking-based algorithm
  - Employs the Shortest-Path algorithm to solve VLiM

M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.

Contributions: Algorithms and Tools

- Find a near-optimal VNE solution using Monte Carlo Tree Search
- Improve the solution by parallelizing MCTS
- Why MCTS?
  - success of MCTS in solving optimization problems
- We developed VNE-Sim:
  - a discrete event VNE simulator written in C++

Ashish Sabharwal, Horst Samulowitz, and Chandra Reddy, "Guiding Combinatorial Optimization with UCT," in *Lecture Notes in Computer Science: Proc. CPAIOR 2012*, N. Beldiceanu, N. Jussien, and E. Pinson, Eds. Springer, 2006, vol. 7298, pp. 356–361.

(2016, Jan.) VNE-Sim repository. [Online]. Available: https://bitbucket.org/shaeri/vne-sim/.

Monte Carlo Tree Search (MCTS)

- The decision-making agent has access to a generative model
  - generates samples of successor states and rewards given current state and an action:

$$\mathcal{G}(s, a) \rightarrow (s', r)$$

- MCTS employs $\mathcal{G}$ to construct a sparse search tree:
  - deepens the tree in the most promising direction

MCTS Parallelization

- Root parallelization
  - processes do not require to share their trees
  - may be implemented lock free
  - less coordination and communication between the processes
  - superior performance

T. Cazenave and N. Jouandeau, "On the parallelization of UCT," in *Proc. Computer Games Workshop*, H. J. van den Herik, J. W. Uiterwijk, and M. H. Winands, Eds. Universiteit Maastricht, 2007, vol. 5131, pp. 93–101.

G. Chaslot, M. Winands, and H. J. Herik, "Parallel Monte-Carlo tree search," *Lecture Notes in Computer Science: Computers and Games*, H. J. Herik, X. Xu, Z. Ma, and M. Winands, Eds. Springer, 2008, vol. 5131, pp. 60–71.

Proposal: MaVEn algorithms

- MaVEn-M:
  - MCF algorithm for link mapping
- MaVEn-S:
  - a breadth-first search-based shortest-path algorithm for link mapping
- Parallelize the MaVEn algorithms
- Advantage:
  - adjustable execution time
    - more computational power = better performance

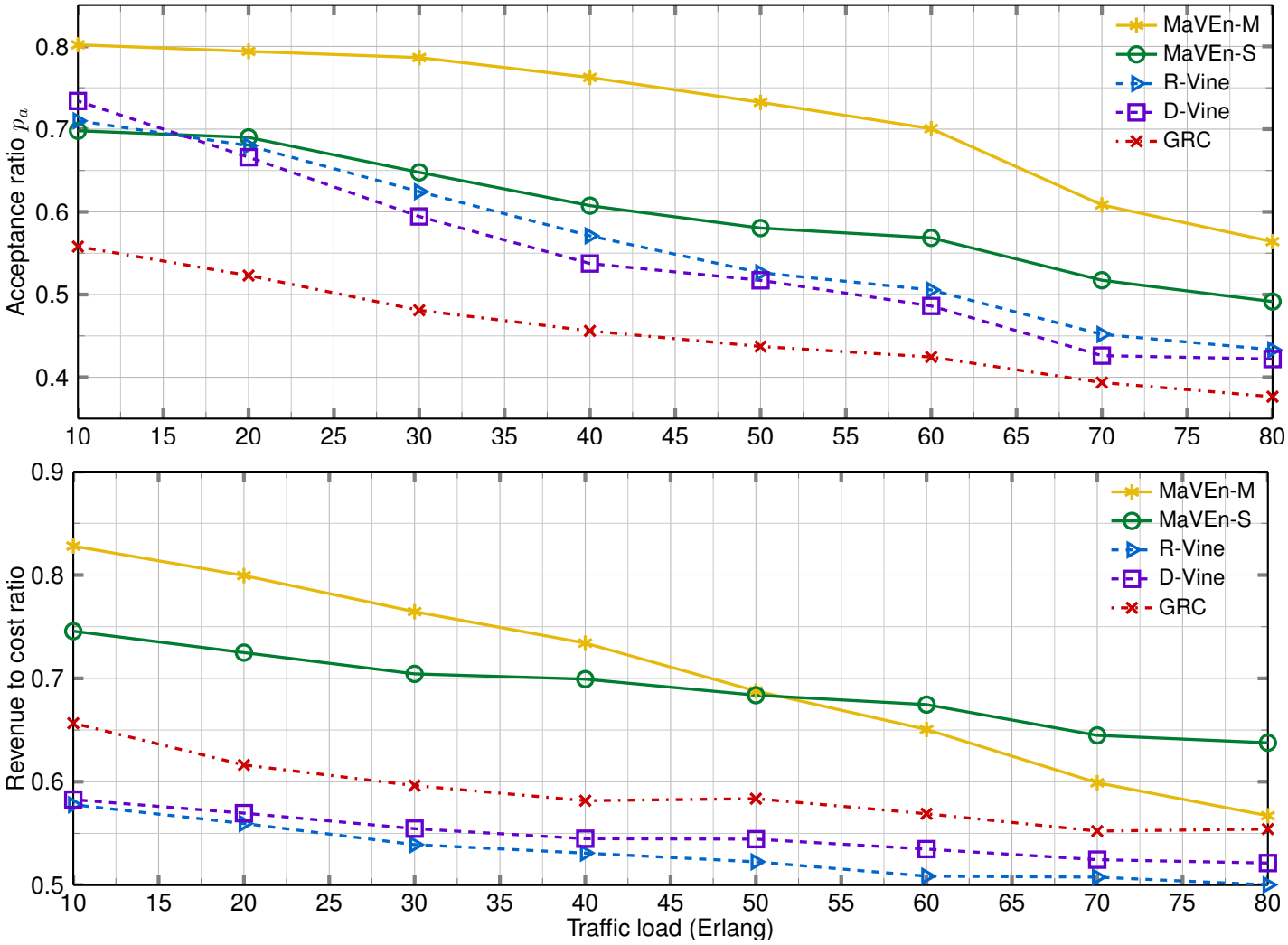
Simulations

- Substrate Network:
  - 50 nodes and 221 edges
- Virtual network requests:
  - number of nodes uniformly distributed between 3 and 10
- Performance measures:
  - acceptance ratio
  - revenue to cost ratio

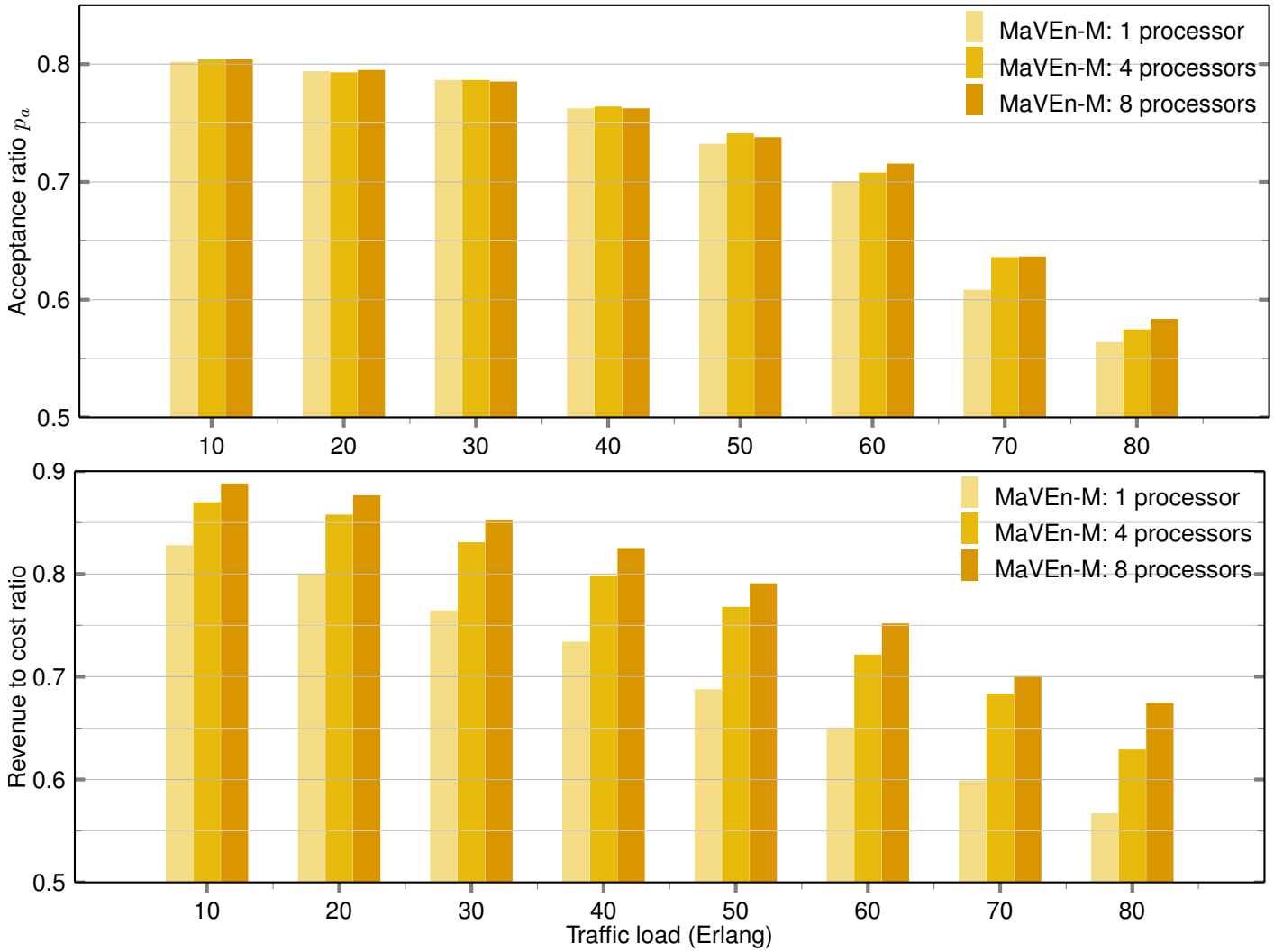
M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.

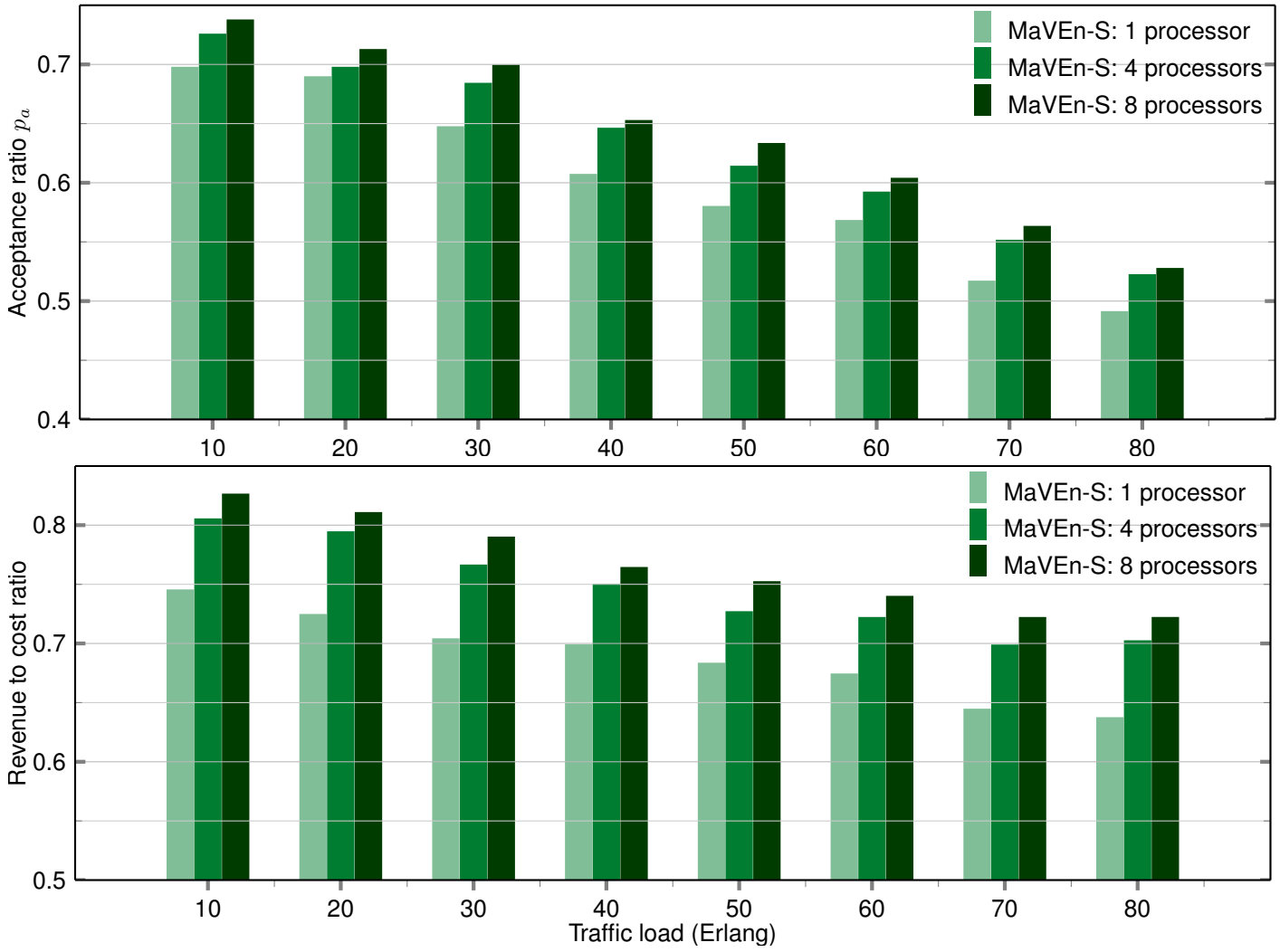
Acceptance and Revenue to Cost Ratios



Parallel MaVEn-M



Parallel MaVEn-S



Discussion

- Proposed algorithms perform better than the existing algorithms
- Superior performance comes at the cost of higher execution time
- The stochastic processes governing distributions of VNR arrivals and life-times have not been well investigated
  - difficult to estimate a reasonable trade-off between execution time and profitability
- Parallelizing MCF and eliminating disk I/O operations improves the execution time of MaVEn-M and Vine Algorithms

Roadmap

- Introduction
- Reinforcement learning
- Deflection routing
- Virtual network embedding
- Conclusion
- Publications and references

Conclusion

- Considered application of reinforcement learning to deflection routing and virtualization in computer networks
- While reinforcement learning algorithms have found applications in various fields, they are still not widely employed for computer networking applications
- Presented algorithms demonstrated that learning algorithms provide viable solutions and may be implemented in computer networks

Roadmap

- Introduction
- Deflection routing
- Virtual network embedding
- Conclusion
- Publications and references

Publications

Journal Article:

- S. Haeri and Lj. Trajković, "Intelligent deflection routing in buffer-less networks," *IEEE Tran. Cybern.*, vol. 45, no. 2, pp. 316–327, Feb. 2015.

Book Chapter

- S. Haeri and Lj. Trajković, "Deflection routing in complex networks," in *Complex Systems and Networks*, J. Lu, X. Yu, G. Chen, and W. Yu, Eds., Berlin: Springer, 2015, ch. 15, pp. 395–422.

Conference Proceedings

- S. Haeri, Q. Ding, Zh. Li, and Lj. Trajković, "Global resource capacity algorithm with path splitting for virtual network embedding," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2016)*, Montreal, QC, Canada, to appear, May 2016.
- S. Haeri and Lj. Trajković, "Virtual network embeddings in data center networks," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2016)*, Montreal, QC, Canada, to appear, May 2016.

Publications

- S. Haeri, R. Gill, M. Hay, T. Wong, and Lj. Trajković, "Multihoming with Locator/ID Separation Protocol: An experimental testbed," in *Proc. The IEEE/IFIP Int. Symp. Integrated Netw. Management (IM 2015)*, Ottawa, ON, Canada, May 2015, pp. 1238–1241.
- Y. Li, H. J. Xing, Q. Hua, X.-Z. Wang, P. Batta, S. Haeri, and Lj. Trajković, "Classification of BGP anomalies using decision trees and fuzzy rough sets," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern. (SMC 2014)*, San Diego, CA, USA, Oct. 2014, pp. 1312–1317.
- S. Haeri and Lj. Trajković, "Deflection routing in complex networks," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2014)*, Melbourne, Australia, June 2014, pp. 2217–2220.
- S. Haeri, M. Arianezhad, and Lj. Trajković, "A predictive q-learning-based algorithm for deflection routing in buffer-less networks," in Proc. *IEEE Int. Conf. Syst., Man, and Cybern. (SMC 2013)*, Manchester, UK, Oct. 2013, pp. 764– 769.
- S. Haeri, W. W.-K. Thong, G. Chen, and Lj. Trajković, "A reinforcement learning-based algorithm for deflection routing in optical burst-switched networks," in *Proc. The 14th IEEE Int. Conf. Inform. Reuse and Integration (IRI 2013)*, San Francisco, USA, Aug. 2013, pp. 474–481.

Publications

- N. Al-Rousan, S. Haeri, and Lj. Trajković, "Feature selection for classification of BGP anomalies using Bayesian models," in Proc. *ICMLC 2012*, Xi'an, China, Jul. 2012, pp. 140–147.
- S. Haeri, D. Kresic, and Lj. Trajković, "Probabilistic verification of BGP convergence," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP 2011)*, Vancouver, BC, Canada, Oct. 2011, pp. 127–128.

References

Optical Burst Switching:

- C. Qiao and M. Yoo, "Optical burst switching (OBS)—a new paradigm for an optical Internet," *J. of High Speed Netw.*, vol. 8, no. 1, pp. 69–84, Mar. 1999.
- A. Zalesky, H. Vu, Z. Rosberg, E. W. M. Wong, and M. Zukerman, "Modelling and performance evaluation of optical burst switched networks with deflection routing and wavelength reservation," in *Proc. INFOCOM*, Hong Kong SAR, China, Mar. 2004, vol. 3, pp. 1864–1871.
- X. Yu, J. Li, X. Cao, Y. Chen, and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *J. Lightw. Technol.*, vol. 22, no. 12, pp. 2722–2738, Dec. 2004.

Reinforcement Learning and Routing:

- L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, USA, May 2002, vol. 2, pp. 1825–1830.
- A. Nowe, K. Steenhaut, M. Fakir, and K. Verbeeck, "Q-learning for adaptive load based routing," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, San Diego, CA, USA, Oct. 1998, vol. 4, pp. 3965–3970.

References

- J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: a reinforcement learning approach," in *Advances in Neural Inform. Process. Syst.*, vol. 6, pp. 671–678, 1994.
- S. P. M. Choi and D. Yeung, "Predictive Q-routing: a memory-based reinforcement learning approach to adaptive traffic control," in *Advances in Neural Inform. Process. Syst.*, vol. 8, pp. 945–951, 1998.
- Y. Kiran, T. Venkatesh, and C. Murthy, "A reinforcement learning framework for path selection and wavelength selection in optical burst switched networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 18–26, Dec. 2007.
- A. Belbekkouche, A. Hafid, and M. Gendreau, "Novel reinforcement learning-based approaches to reduce loss probability in buffer-less OBS networks," *Comput. Netw.*, vol. 53, no. 12, pp. 2091–2105, Aug. 2009.

References

Network Virtualization:

- M. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20–26, July 2009.
- N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, Jan. 2007.

Virtual Network Embedding Algorithms:

- L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.
- M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.
- S. Zhang, Y. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2408–2416.

References

- X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *Comput. Commun. Rev.*, vol. 41, pp. 38–47, Apr. 2011.
- M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 19–29, Mar. 2008.

Introduction: Proposal

- Two applications of reinforcement learning algorithms in computer networks:
  - deflection routing in optical burst-switched networks

  and
  - virtual network embedding

Reinforcement Learning

- Formalizes trial-and-error-based learning processes
- Four basic elements:
  - an agent or decision maker
  - the environment of the agent
  - actions that the agent performs
  - feedback signals generated by the environment

Markov Decision Process (MDP)

- MDP is the quintuple: $(\mathcal{T}, \mathcal{S}, \mathcal{A}, R, p)$
  - $\mathcal{T}$: set of all decision-making instances
  - $\mathcal{S}$: state space
  - $\mathcal{A}$: action space
  - $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ reward function
    - assigns real-valued rewards to state-action pairs
  - $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ transition probability distribution
    - defines the distribution of the next state given a state and an action

Markov Decision Process (MDP)

- Objective:
  - find an action policy π that maximizes the expected reward
- Exact MDP solution:
  - value and policy iteration algorithms
  - polynomial in the size of state space
  - infeasible for MDP with exponentially large state space

M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. NJ, USA: John Wiley & Sons, 1994.

Neural Networks for Deflection Routing

- Feed-forward neural network for generating deflection decisions consists of:
  - State: input layer $\mathbf{I} = \left[\mathbf{I}^l \;\; \mathbf{I}^d\right]$
  - Middle layer $\mathbf{Z}^m = [z_1^m \; \ldots \; z_n^m]$
  - Deflection decision: output layer $\mathbf{Z}^o = [z_1^o \; \ldots \; z_n^o]$

Weight Updates

Neural network:

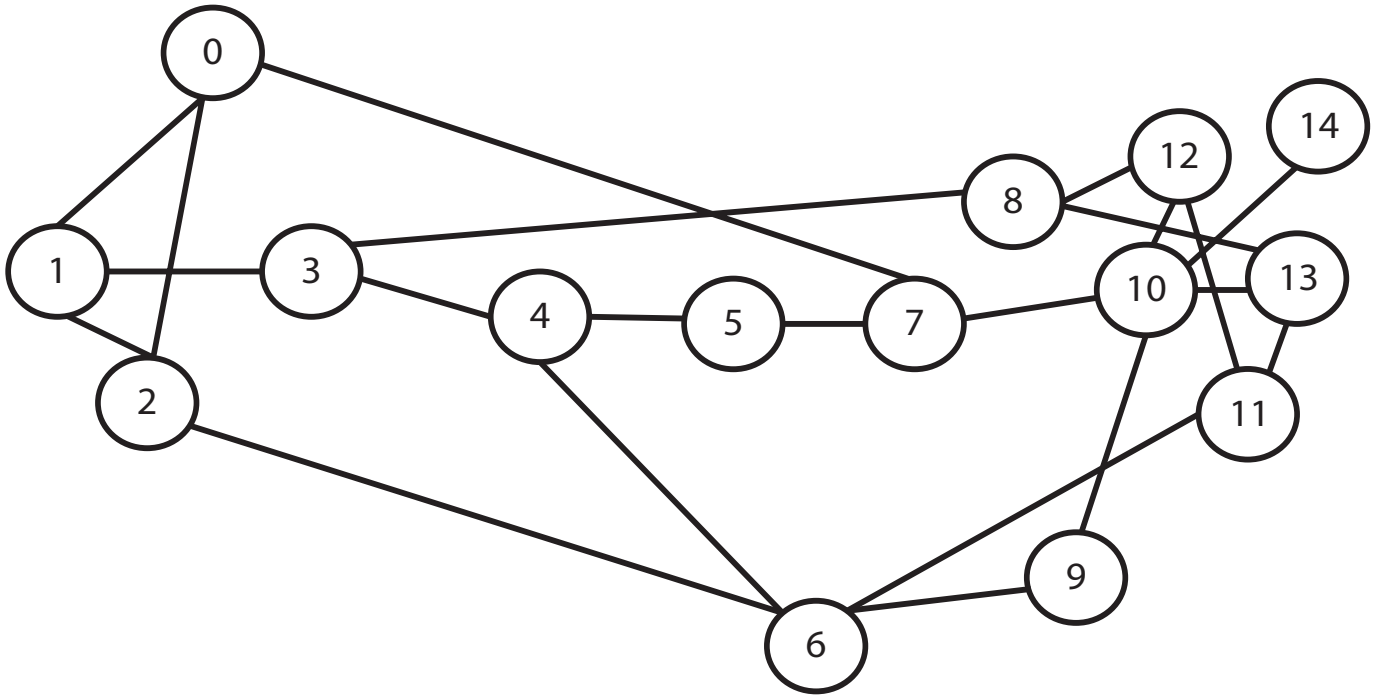
- Receives a state $\mathbf{I}_t$
- Translates the state to a deflection decision $\mathbf{Z}_t^o$
- Waits for the reward signal r
- Updates its weight matrices:

$$\Delta w_{ij} = \alpha r (y_i - p_i) x_{ij}$$

  - α: non-negative rate factor
  - y_i: output of the ith neuron
  - p_i: probability of $y_i = 1$, given the input and weight vectors

Simulations: National Science Foundation Network

- National Science Foundation (NSF) network topology:

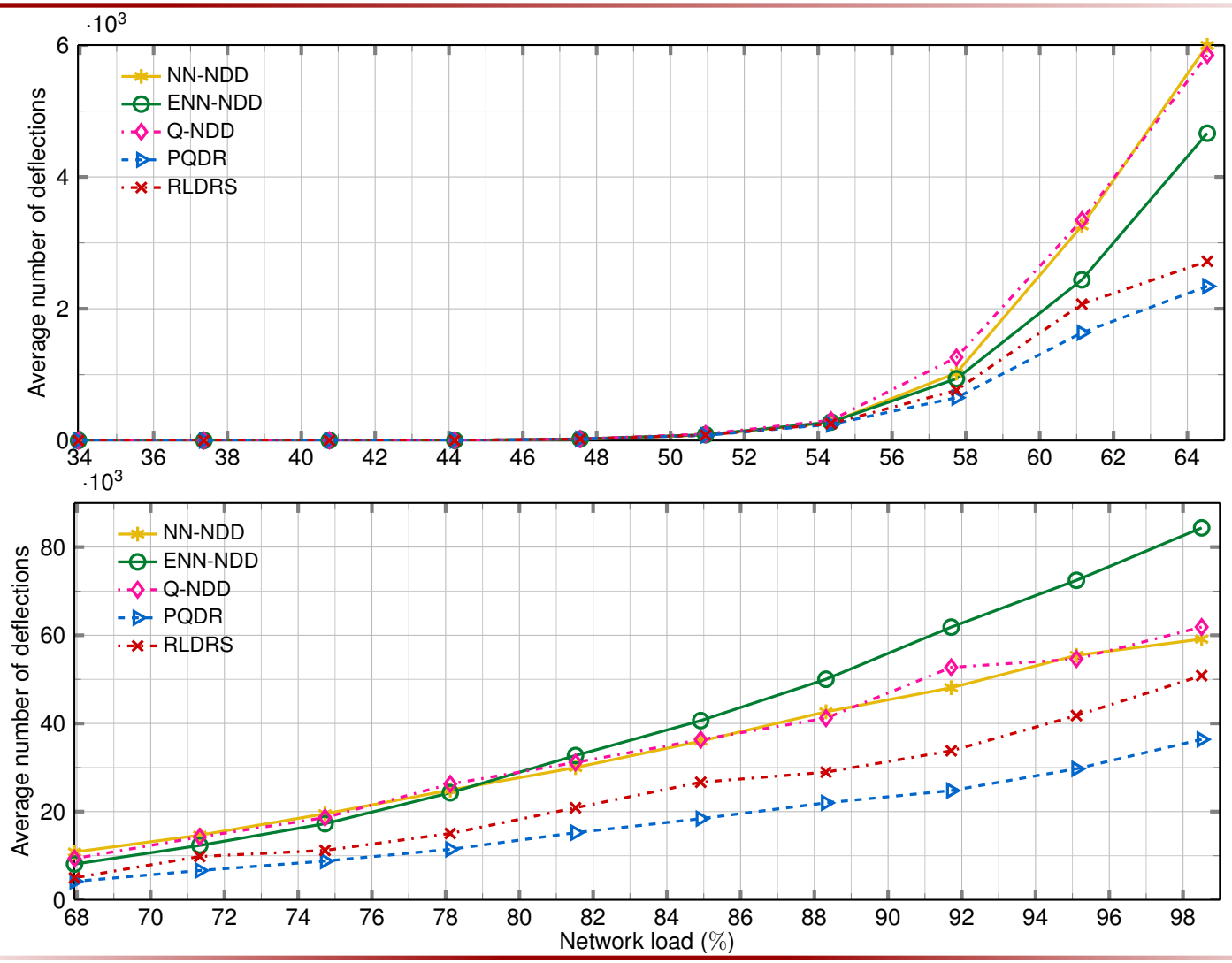


NSF network after the 1989 transition

Simulations: Setup

- Buffer-less optical burst switching architecture:
  - 1 Gbps fiber links
  - 64 wavelengths
- Traffic flows:
  - Poisson arrivals
  - 0.5 Gbps data rate
  - 50 bursts:
    - each burst carries 12.5 kB payload

Results: Average Number of Deflections



Substrate vs. Virtual Networks

- InPs operate physical substrate networks (SNs)
- SN components:
  - physical (substrate) nodes
  - physical (substrate) links
- Substrate nodes and links are:
  - interconnected using arbitrary topology
  - used to host various virtualized networks with arbitrary topologies
- Virtual networks are embedded into a substrate network

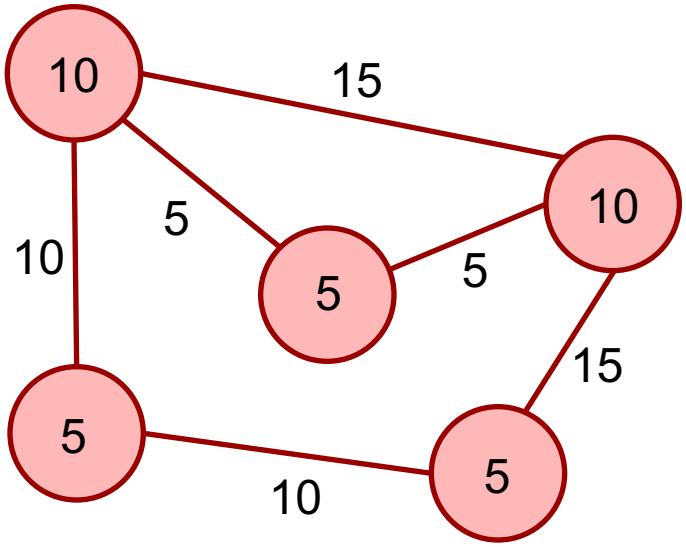
VNE Solution: VLiM and Path Splitting

- Path splitting:
  - a flow may be divided into multiple flows with lower capacity
- The shortest-path algorithms do not permit path splitting:
  - stricter than the MCF algorithm
- MCF enables path splitting:
  - flows are routed through various paths

D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, Unpublished Manuscript. [Online]. Available: http://repository.cmu.edu/compsci/86/.

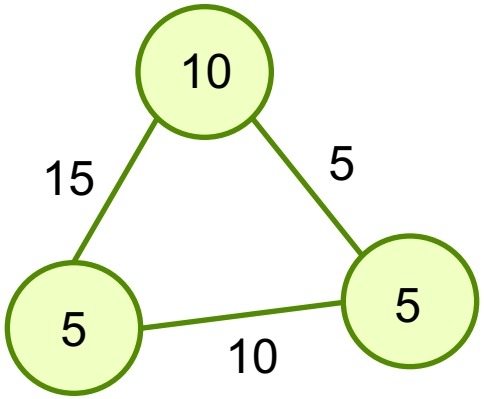
VNE Formulation: Constrains

- Substrate network graph: $G^s(N^s, E^s)$
- Resources:
  - substrate nodes: CPU capacity $\mathcal{C}(n^s)$
  - substrate links: bandwidth $\mathcal{B}(e^s)$



VNE Formulation: Constrains

- Virtual network graph: $G^{\Psi_i}(N^{\Psi_i}, E^{\Psi_i})$
- Resources:
  - virtual nodes: CPU capacity $\mathcal{C}(n^{\Psi_i})$
  - virtual links: bandwidth $\mathcal{B}(e^{\Psi_i})$



VNE as a Markov Decision Process

- Consider a substrate network with j nodes and k links
- At an arbitrary time instant ω, the VNE solver receives for embedding a VNR Ψ_i that requires ℓ virtual nodes and m virtual links.
- First solve VNoM:
  - select ℓ substrate node for embedding VNR nodes
  - yields ℓ decision-making instances (horizon)

MDP: Initial State and Action Set

- Initial state: $s_1 = (N_1^{\Psi_i}, N_1^s)$
  - $N_1^{\Psi_i}$: ordered set of all virtual nodes yet to be embedded
  - N_1^s : set of all substrate nodes initially available for embedding
- Initial action set:

$$\mathcal{A}_1^{\Psi_i} = \{\varepsilon\} \cup \left\{(n_1^{\Psi_i}, n^s) : \forall n^s \in \{N^s(n_1^{\Psi_i}) \cap N_1^s\}\right\}$$

  - ε : forced transition to terminal state
  - $N^s(n_1^{\Psi_i})$: set of all substrate nodes with enough resources for embedding $n_1^{\Psi_i}$

MDP: State Transition

- The decision-making agent selects a substrate node n_a^s for embedding $n_1^{\Psi_i}$
- The MDP transitions to state:

$$s_2 = \left(N_2^{\Psi_i} = N_1^{\Psi_i} \setminus \{n_1^{\Psi_i}\}, N_2^s = N_1^s \setminus \{n_a^s\}\right)$$

  - with probability 1
- The action set in the second time step:

$$\mathcal{A}_2^{\Psi_i} = \{\varepsilon\} \cup \left\{(n_2^{\Psi_i}, n^s) : \forall n^s \in \{N^s(n_2^{\Psi_i}) \cap N_2^s\}\right\}$$

MDP: Reaching the Horizon

- The decision-making horizon is reached after selecting ℓ substrate nodes for embedding virtual nodes
- The agent proceeds to identifying the virtual link mappings (solve VLiM)
- If VLiM is successful, the agent receives a terminal reward

$$R_\rho = \mathbf{R}(G^{\Psi_i}) - \mathbf{C}(G^{\Psi_i})$$

- Else, the reward:

$$R_\rho = \Gamma$$

MDP: Finding the Optimal Policy

- Number of substrate and virtual nodes:
  - determine the number of MDP state variables
  - exponential growth of number of states
- Finding an exact solution for the MDP is intractable for even fairly small substrate and virtual networks

Monte Carlo Tree Search (MCTS)

- The nodes and edges of the search tree correspond to states and actions, respectively
- Each tree node stores a value v and a visit count σ
- MCTS algorithm begins with a tree that only consists of the root node
- Executes four phases until a predefined computational budget is exhausted:
  - selection
  - expansion
  - simulation
  - backpropagation

MCTS: Selection Phase

- The tree is traversed from the root until a non-terminal leaf node is reached
- A child node is selected based on a selection strategy at each level of the tree
- Exploration vs. Exploitation dilemma
  - explore the undiscovered sections of the search tree to find better actions

  or

  - exploit promising subtrees that have already been discovered

Upper Confidence Bound for Trees

- One of the most commonly used selection strategies:

$$\kappa \in \arg\max_{i \in \mathcal{I}} \left(\frac{v_i}{\sigma_i} + D\sqrt{\frac{\ln \sigma_u}{\sigma_i}} \right)$$

- κ : selected child
- u : current node
- $\mathcal{I}$: set of all children of u
- D: exploration constant

L. Kocsis and C. Szepesvari, "Bandit based Monte-Carlo planning," in *Lecture Notes in Computer Science: Proc. 17th European Conf. on Machine Learning (ECML)* Springer, 2006, vol. 4212, pp. 282–293.

MCTS: Expansion

- After a non-terminal leaf node is selected, one or more of its successors are added to the tree
- The most common expansion strategy is to add one node for every execution of the four MCTS phases
- The new node corresponds to the next state

R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo Tree Search," in *Proc. 5th Int. Conf. Comput. and Games (CG'06)*, Turin, Italy, May 2006, pp. 72–83.

MCTS: Simulation

- The generative model $\mathcal{G}$ is used to perform a sequence of actions from non-terminal until a terminal node is reached
- MCTS converges if actions are randomly selected
- Utilizing domain knowledge may improve the convergence speed

MCTS: Backpropagation

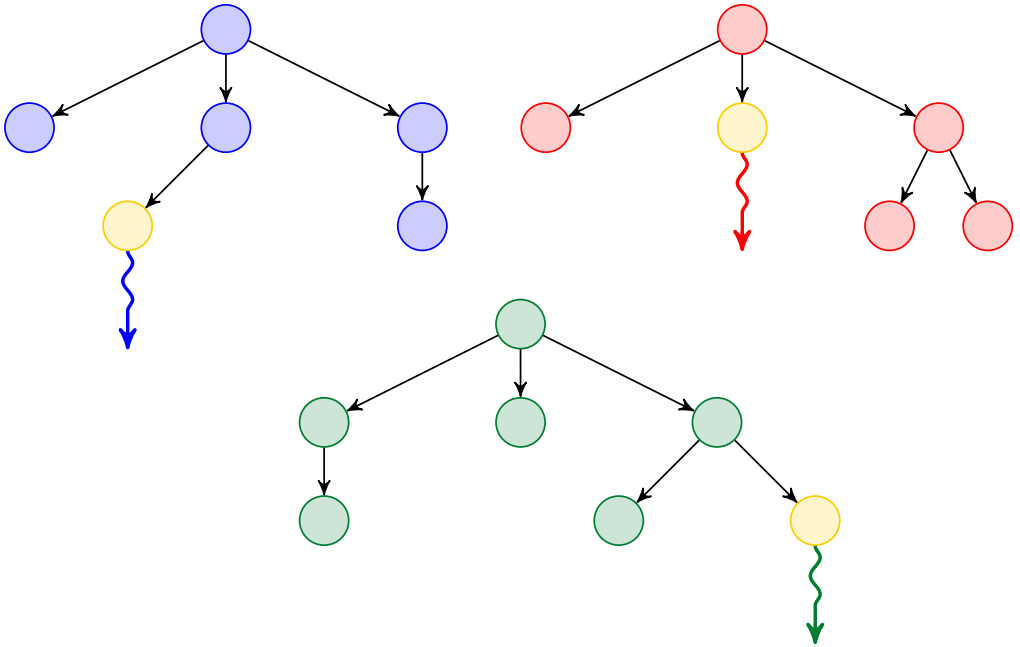
- The reward is calculated after a leaf node is reached in the Simulation phase
- Reward is then propagated from the leaf node to the root
- Every tree node in the current trajectory is updated:
  - adding reward to its current value v
  - incrementing its count σ

MCTS Parallelization

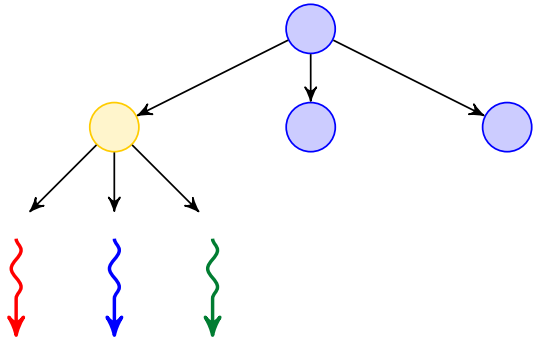
Common techniques:

- Leaf parallelization
- Root parallelization
  - processes do not require to share their trees
  - may be implemented lock free
  - less coordination and communication between the processes
  - superior performance

MCTS Parallelization



Root Parallelization

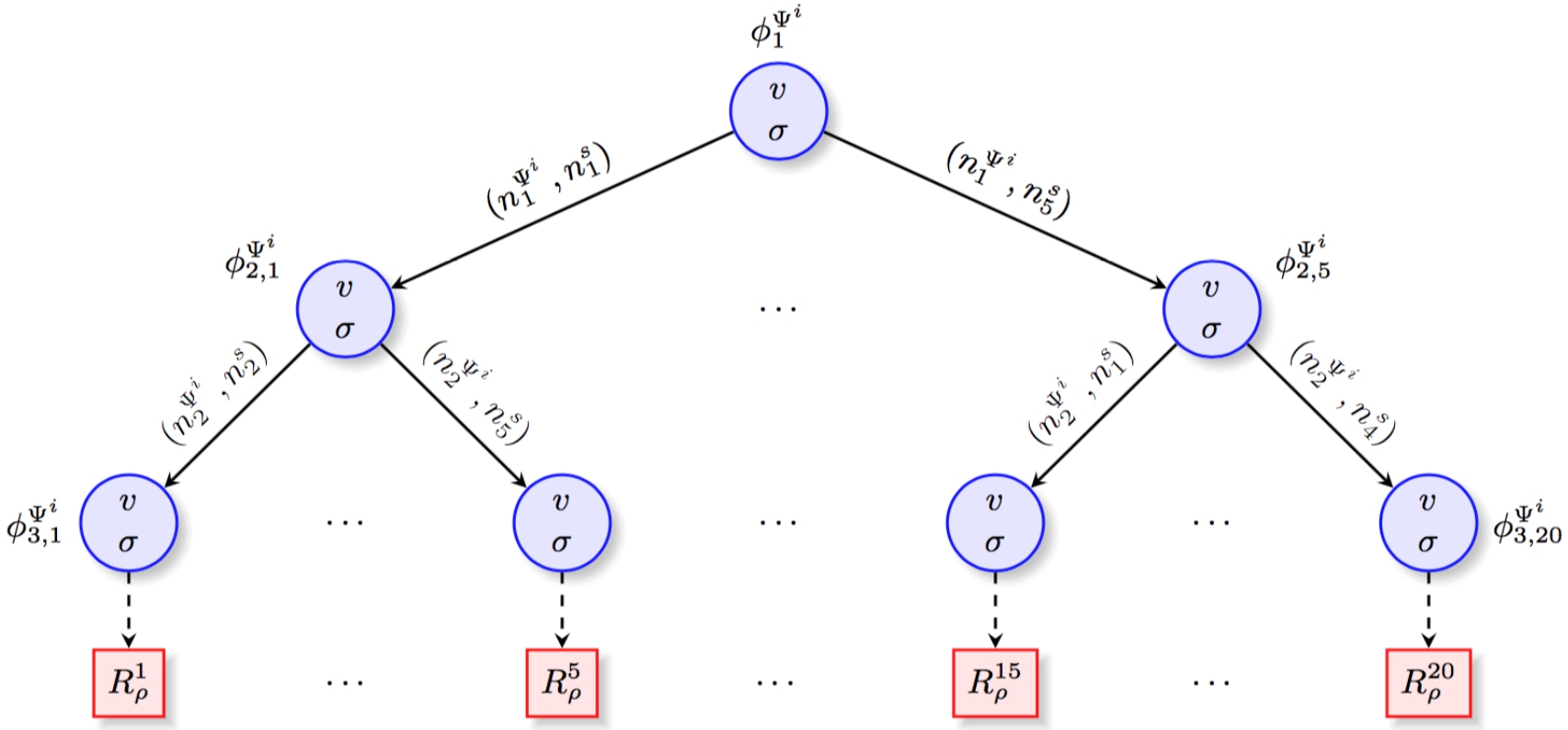


Leaf Parallelization

Search tree node of Proccess #1
Search tree node of Proccess #2
Search tree node of Proccess #3
Search tree node selected by UCT

Current location of Proccess #1
Current location of Proccess #2
Current location of Proccess #3

VNE Search Tree Example



Search tree for embedding a VNR with 3 nodes onto a substrate network with 5 nodes

Simulations

- Substrate Network:
  - 50 nodes and 221 edges
  - Nodes' CPU and links' bandwidth capacities uniformly distributed between 50 and 100 units
- Virtual network requests:
  - number of nodes uniformly distributed between 3 and 10
  - CPU requirements uniformly distributed between 2 and 20
  - bandwidth requirements uniformly distributed between 0 and 50

Simulations: MCTS Parameters

- Computational budget: 40
- Exploration constant: 0.5

M. P. D. Schadd, M. H. M. Winands, M. J. W. Tak, and J. W. H. M. Uiterwijk, "Single-player Monte-Carlo tree search for SameGame," *Knowledge-Based Systems*, vol. 34, pp. 3–11, Oct. 2012.

Simulations: Other Parameters

- MT19937 Mersenne Twister random number generator with seed 0
- Poisson distribution for arrivals with means 1 to 8 units per 100 time units
- Exponentially distributed life-times with mean 1,000 time units
- Traffic loads: 10, 20, 30, 40, 50, 60, 70, and 80 Erlangs
- Total simulation time: 50,000 time units
- Performance metrics:
  - acceptance ratio, revenue to cost ratio

Discussion

- Parallelizing MCF and eliminating disk I/O operations improves the execution time of MaVEn-M and Vine Algorithms
- MCTS parallelization improves performance of the MaVEn algorithms
  - may be executed on clusters that are highly optimized for parallel computing and comprise large number of processors