Relational Deep Reinforcement Learning for Routing in Wireless Networks

Victoria Manfredi, Alicia Wolfe, Bing Wang, Xiaolan Zhang

WoWMoM

June 9, 2021

Outline

1. Motivation
2. Our DeepRL approach
3. Evaluation
4. Related Work
5. Wrap-up

Outline

1. **Motivation**
2. Our DeepRL approach
3. Evaluation
4. Related Work
5. Wrap-up

Multi-hop wireless network

Devices

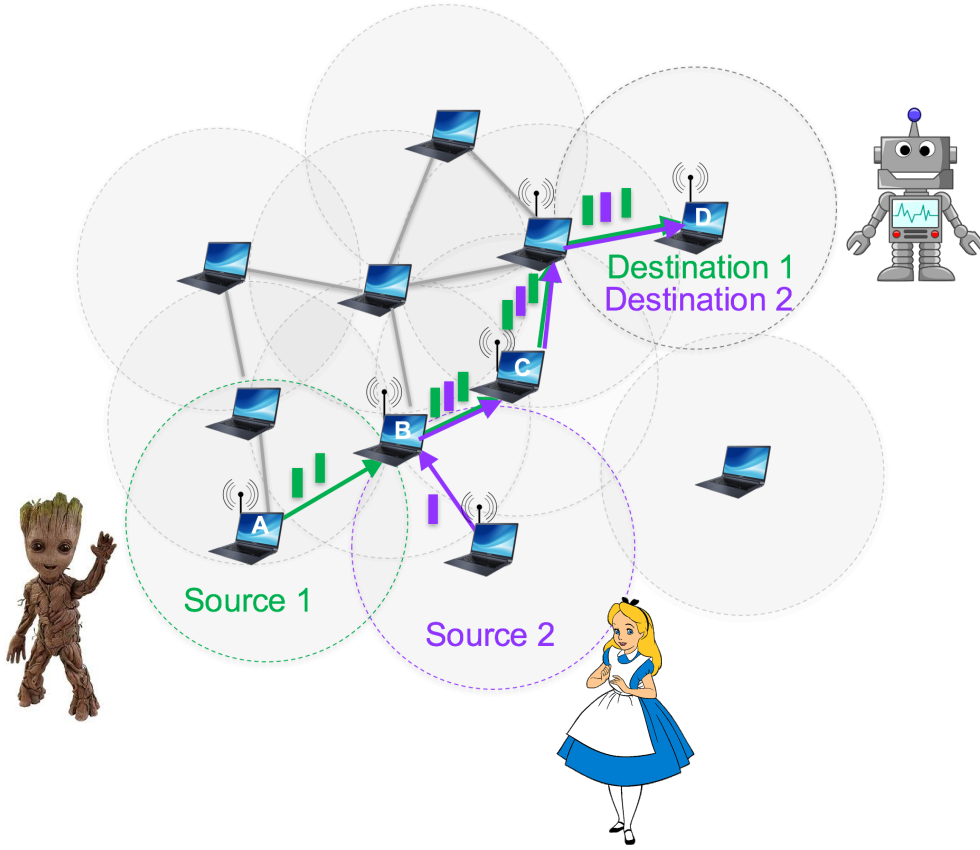
- operate as both end-hosts and routers (forward traffic)

Why multi-hop?

- ease of deployment (no infrastructure needed), privacy

Problem: routing is hard

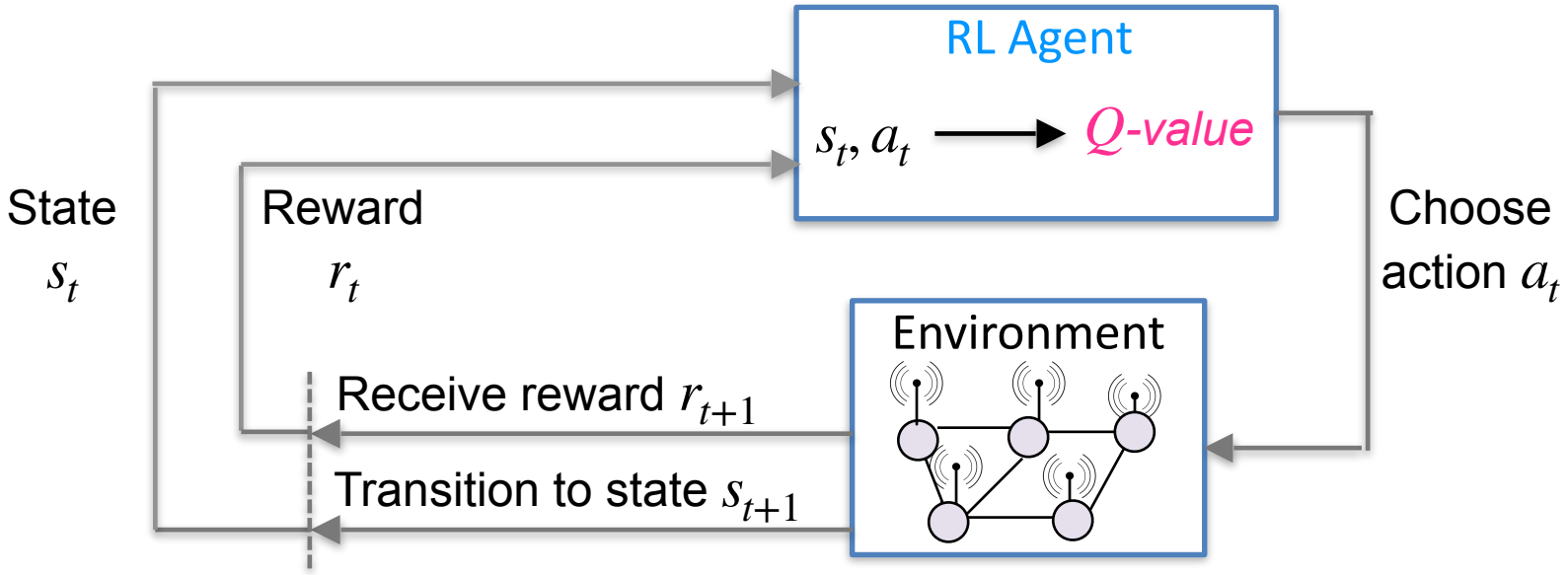
- **changing network conditions**: traffic, connectivity, interference, mobility
- **competing routing goals**: throughput, delay, power



Solution: learn how to route using deep reinforcement learning

What is deep reinforcement learning?

RL agent learns to choose actions to maximize expected future reward



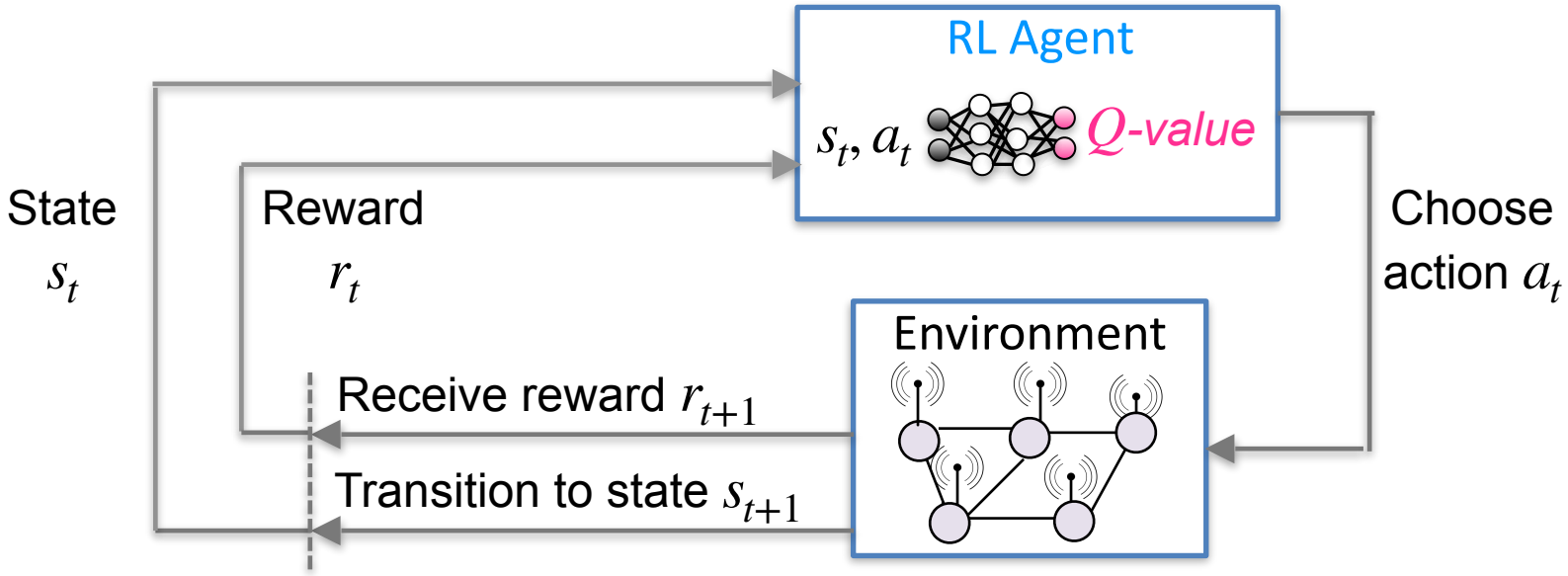
Simple update rule

$$Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t, a_t) + \alpha \left[r + \gamma \max_{a_t} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

Move estimate closer to target

What is deep reinforcement learning?

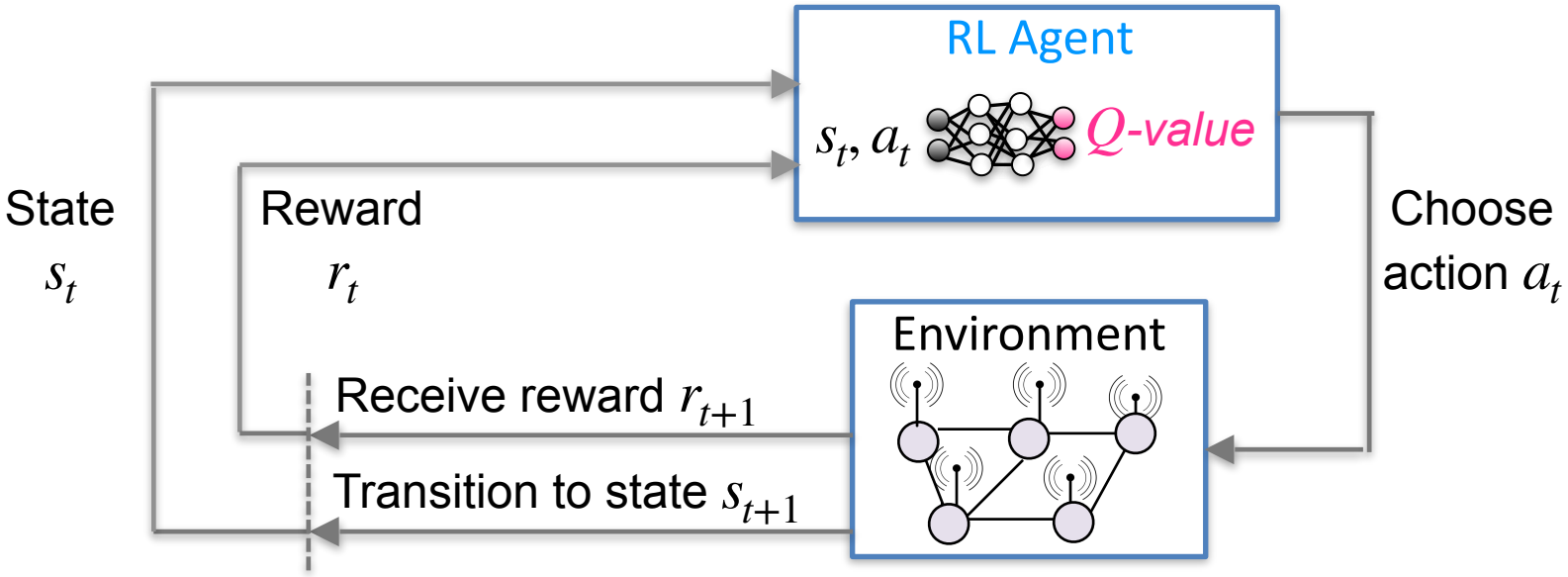
RL agent learns to choose actions to maximize expected future reward



Use deep neural network to approximate mapping from (state s_t, action a_t) to Q-value

What is deep reinforcement learning?

RL agent learns to choose actions to maximize expected future reward



Our goal: define RL agent for routing. Requires us to define ***states***, ***actions***, and ***rewards*** useful for routing

Outline

1. Motivation

2. **Our DeepRL approach**
   - **Who chooses actions? I.e., who should be a DeepRL agent?**
   - How to define generalizable states and actions?
   - Can we make training more efficient?

3. Evaluation

4. Related Work

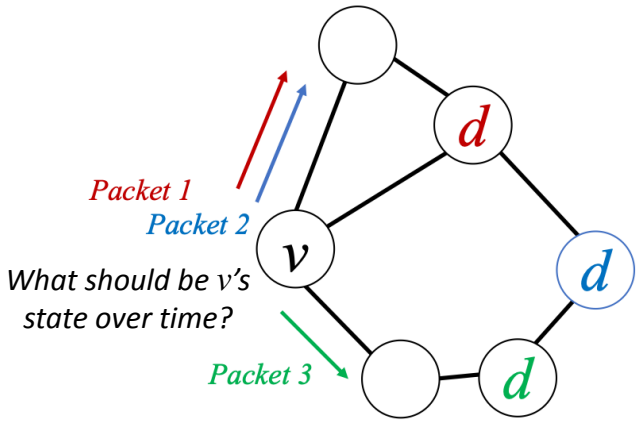
5. Wrap-up

Who chooses actions?

Problem: normally a device chooses a packet's next hop. But a device's state doesn't track what happens with a forwarded packet

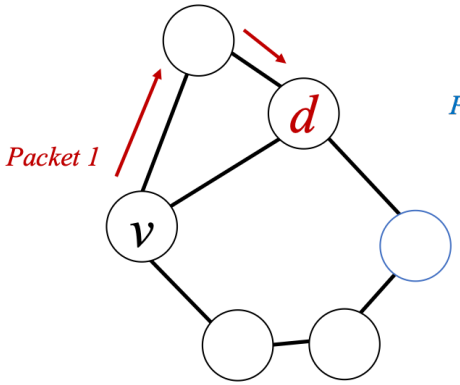
Device agent

Device v chooses next hop for outgoing packet

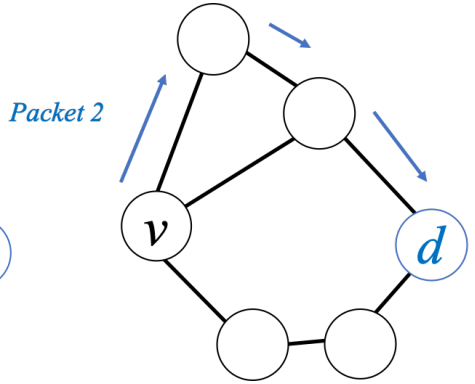


Packet agent

Packet 1 chooses next hop at each device



Packet 2 chooses next hop at each device



Solution*:* use packet agents. Simplifies ***experience sequence of*** s, a, s', r , and ***easily defines reward*** for packet drops, deliveries, forwards, queueing

Outline

1. Motivation

2. **Our DeepRL approach**
   - Who chooses actions? I.e., who should be a DeepRL agent?
   - **How to define generalizable states and actions?**
   - Can we make training more efficient?

3. Evaluation

4. Related Work

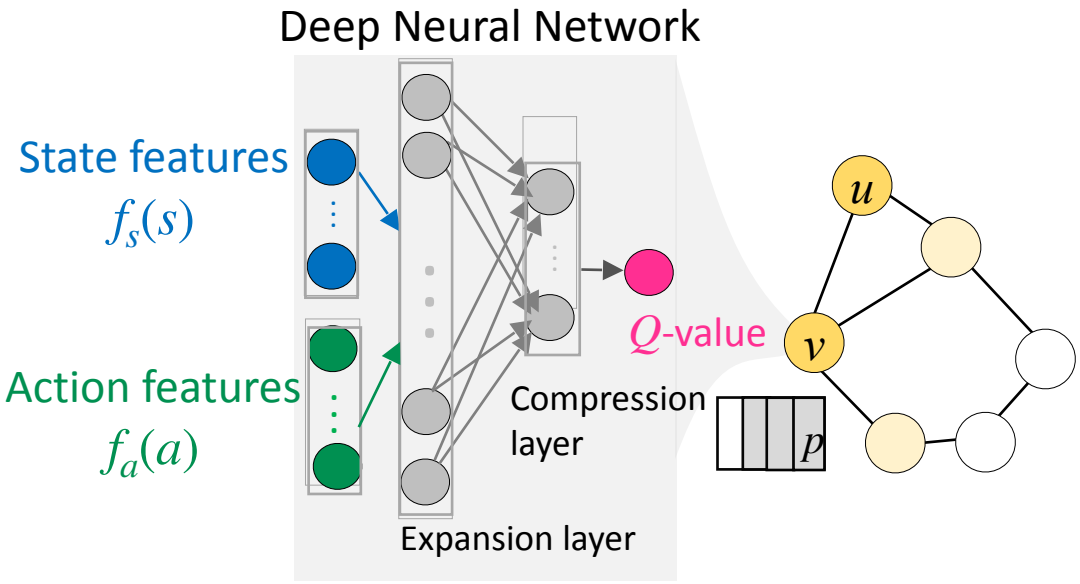
5. Wrap-up

How to define generalizable states and actions?

Problem: network connectivity varies, but # of inputs to DNN are fixed

How packet p at device v chooses next hop

- For each (state, action) pair, inputs features into DNN to get Q-value
- p chooses action that gives highest Q-value

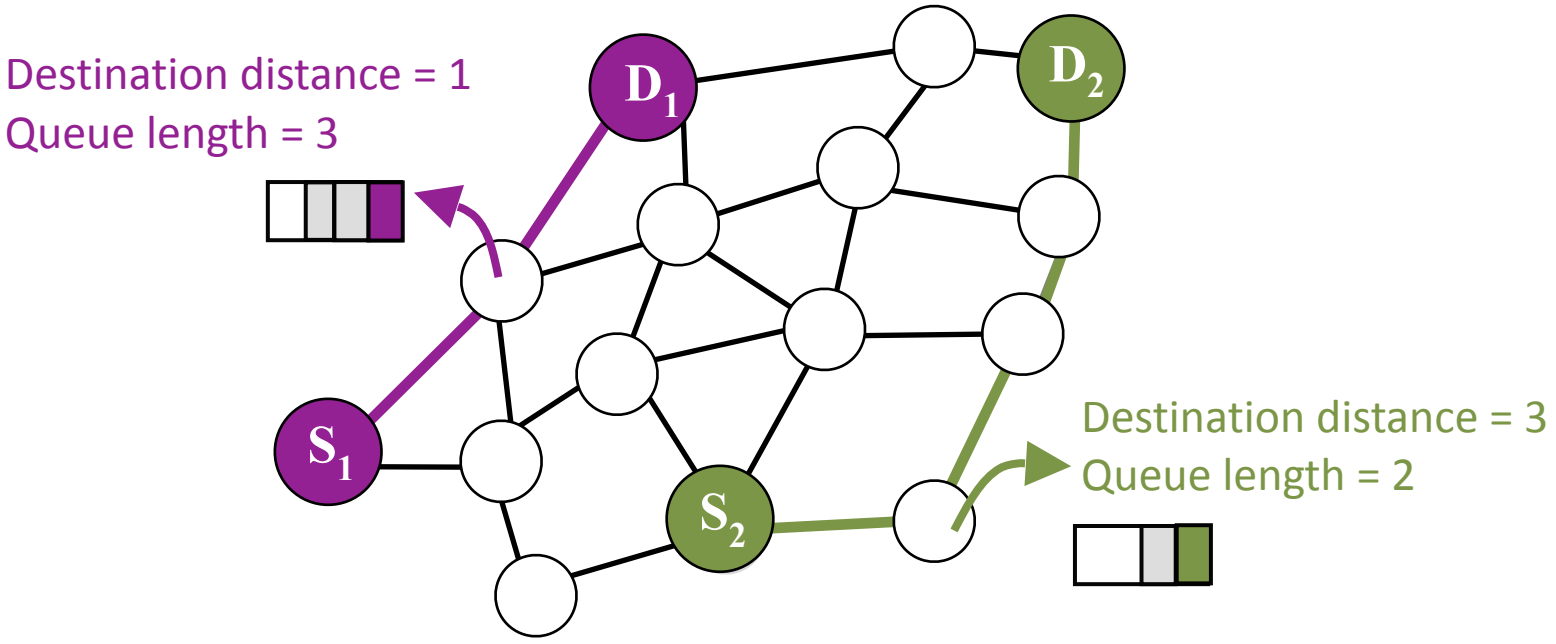


Goal 1: be able to use the same DNN for topologies with different connectivity
Goal 2: handle varying # of next hop actions despite DNNs having fixed # of inputs

Relational features support generalizability

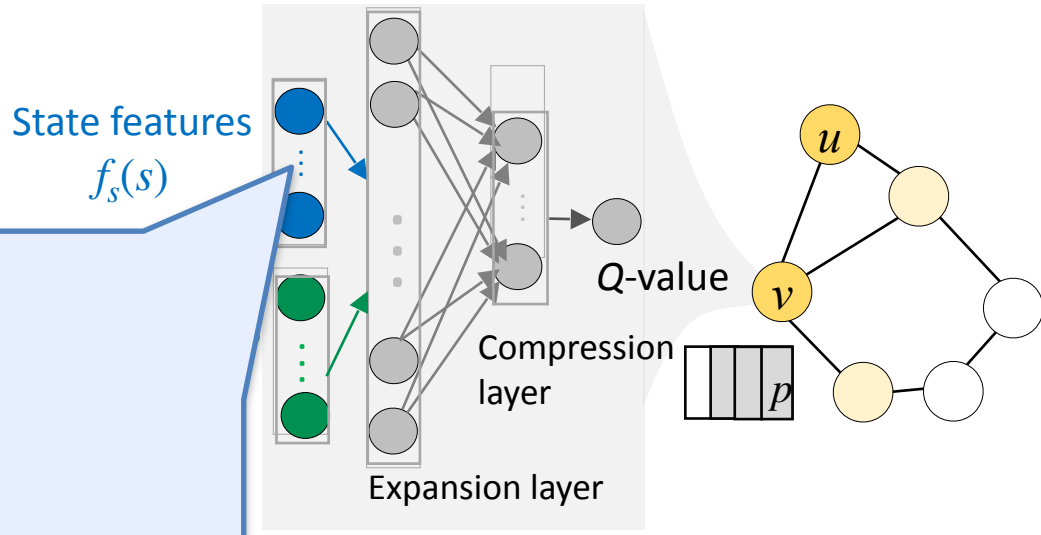
Relational features are independent of network topology and traffic

- ***Relational:*** distance to destination, queue length, …
- ***Not relational***: device ID, packet destination ID, traffic matrix, …



Relational state features

For packet p at device v with 1-hop neighbor set $Nbr(v)$



Packet features $f_{pkt}(p, v, t)$

- p's TTL, p's location in v's queue

Local device features $f_{device}(v, p, t)$

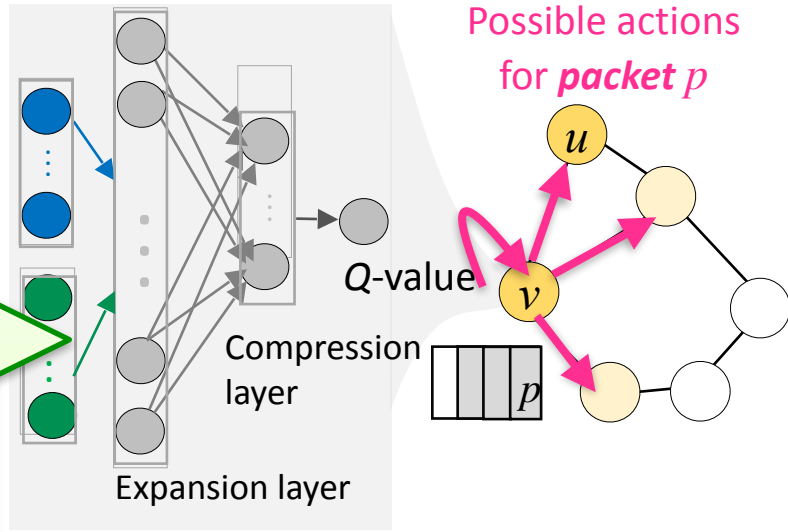
- distance from v to p's dest.
- v's queue length
- v's queue length for only packets to p's dest.
- v's node degree

Aggregated neighbor features, $f_{nbr}(Nbr(v), p, t)$

- summarize varying # of neighbors
- min, mean, max of $f_{device}(Nbr(v), p, t)$

Relational action features

Packet p separately considers each possible action u



Let $u \in Nbr(v) \cup v$. Then action u's features are given by the **local device features** $f_{device}(u, p, t)$

- distance from u to p's dest.
- u's queue length
- u's queue length for only packets to p's dest.
- u's node degree

Outline

1. Motivation

2. **Our DeepRL approach**
   - Who chooses actions? I.e., who should be a DeepRL agent?
   - How to define generalizable states and actions?
   - **Can we make training more efficient?**

3. Evaluation

4. Related Work

5. Wrap-up

Can we make training more efficient?

Problem: training requires lots of computation and data

Solution: offline centralized training, online distributed testing

Benefits:

1. Avoids expending bandwidth or computation on online training
2. Allows data from all DeepRL agents to be used in training, with each DeepRL agent independently using same model during testing

Problem: how to efficiently model ***multi-timestep actions***, such as when a packet arrives in a queue and must wait until it can be forwarded

Solution: use extended-time actions, aka options (MDP becomes semi-MDP)

Benefits:

1. Faster learning with less data needed
2. Actions logically match packet behavior

Outline

1. Motivation
2. Our DeepRL approach
3. **Evaluation**
4. Related Work
5. Wrap-up

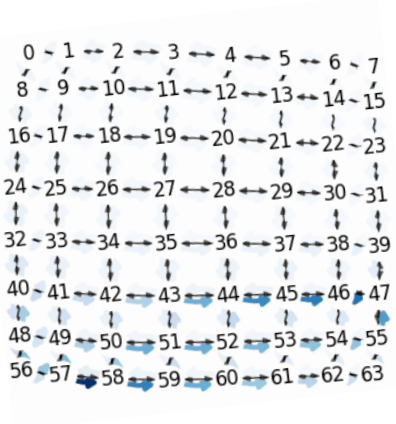
Evaluation overview

Goals:

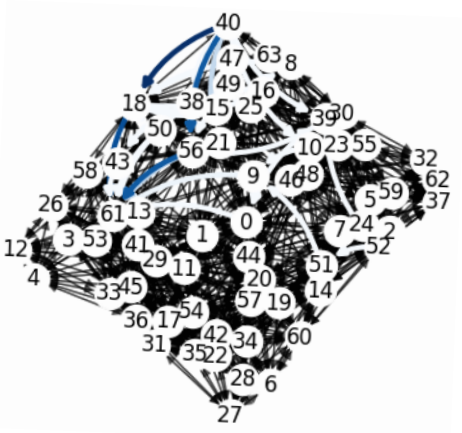
- Identify scenarios for which our DeepRL approach performs well, and
- Test how well a DeepRL agent trained on one scenario is able to generalize its learned routing policy to unseen scenarios

Training and **testing** scenarios:

1. Static lattice + low traffic
2. Static random + high traffic
3. Static lattice + high traffic
4. Dynamic lattice + high traffic
5. Delay tolerant lattice + high traffic
6. Delay tolerant random + high traffic



Lattice



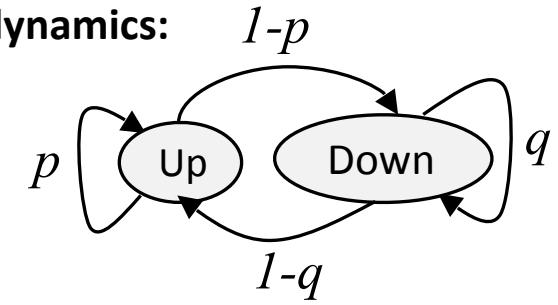
Geometric Random

Evaluation overview

Goals:

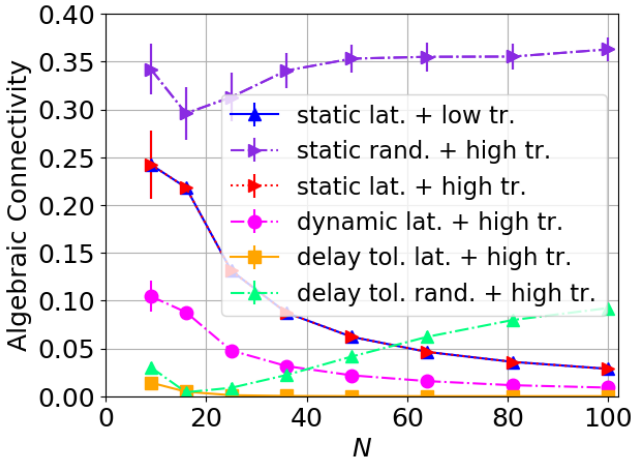
- Identify scenarios for which our DeepRL approach performs well, and
- Test how well a DeepRL agent trained on one scenario is able to generalize its learned routing policy to unseen scenarios

Link dynamics:



Training and **testing** scenarios:

1. Static lattice + low traffic
2. Static random + high traffic
3. Static lattice + high traffic
4. Dynamic lattice + high traffic: $p = .8, q = .2$
5. Delay tolerant lattice + high traffic: $p = .5, q = .4$
6. Delay tolerant random + high traffic: $p = .5, q = .4$



Evaluation overview

Goals:

- Identify scenarios for which our DeepRL approach performs well, and
- Test how well a DeepRL agent trained on one scenario is able to generalize its learned routing policy to unseen scenarios

Training and **testing** scenarios:

1. Static lattice + low traffic
2. Static random + high traffic
3. Static lattice + high traffic
4. Dynamic lattice + high traffic: $p = .8,\ q = .2$
5. Delay tolerant lattice + high traffic: $p = .5,\ q = .4$
6. Delay tolerant random + high traffic: $p = .5,\ q = .4$

Training:

$N = 64$

Testing:

$N = 9,\ 16,\ 25,\ 36,\ 49,\ 64,\ 81,\ 100$

Evaluation overview

Goals:

- Identify scenarios for which our DeepRL approach performs well, and
- Test how well a DeepRL agent trained on one scenario is able to generalize its learned routing policy to unseen scenarios

Training and **testing** scenarios:

1. Static lattice + low traffic
2. Static random + high traffic
3. Static lattice + high traffic
4. Dynamic lattice + high traffic: $p = .8,\ q = .2$
5. Delay tolerant lattice + high traffic: $p = .5,\ q = .4$
6. Delay tolerant random + high traffic: $p = .5,\ q = .4$

Training:

$N = 64$

Testing:

$N = 9,\ 16,\ 25,\ 36,\ 49,\ 64,\ 81,\ 100$

Low traffic: $\lambda_F = .002N/25, \quad \lambda_P = .05, \quad \lambda_D = 5000$

High traffic: $\lambda_F = .002N/25, \quad \lambda_P = .2, \quad \lambda_D = 5000$

Flow arrivals (Poisson) | Flow durations (Exponential) | Packet arrivals (Poisson)

Evaluation overview

Goals:

- Identify scenarios for which our DeepRL approach performs well, and
- Test how well a DeepRL agent trained on one scenario is able to generalize its learned routing policy to unseen scenarios

Training and **testing** scenarios:

1. Static lattice + low traffic
2. Static random + high traffic
3. Static lattice + high traffic
4. Dynamic lattice + high traffic: $p = .8,\ q = .2$
5. Delay tolerant lattice + high traffic: $p = .5,\ q = .4$
6. Delay tolerant random + high traffic: $p = .5,\ q = .4$

Training:

$N = 64$

Testing:

$N = 9,\ 16,\ 25,\ 36,\ 49,\ 64,\ 81,\ 100$

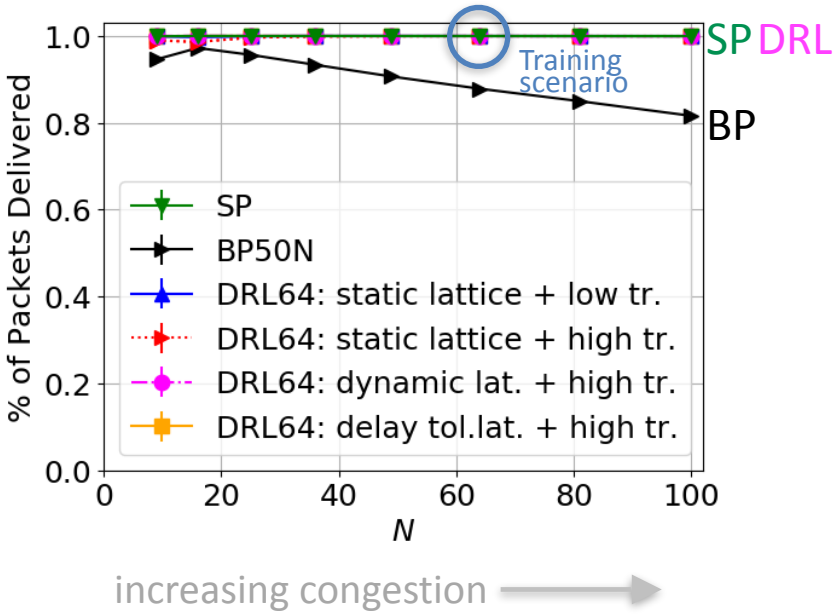
Routing strategies

Shortest Path (SP) vs. Back Pressure (BP) vs. DeepRL agent (DRL)

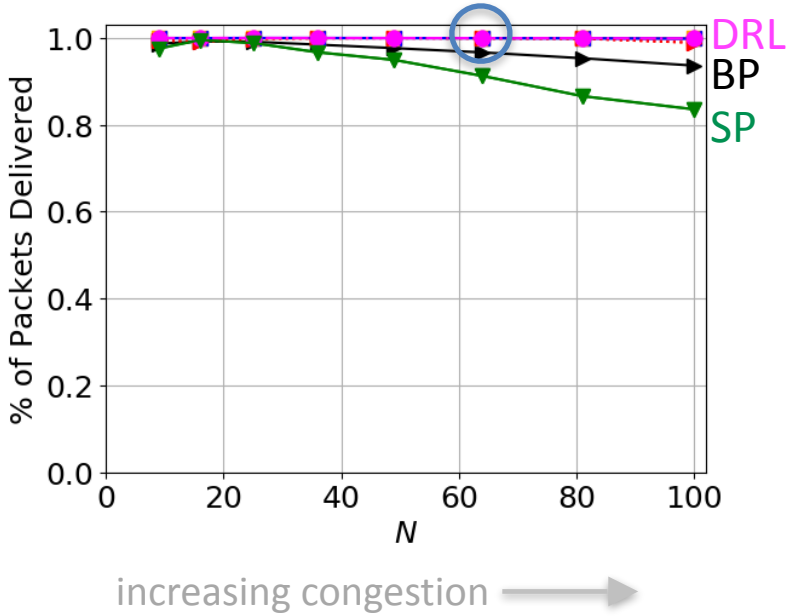
Max queue length is $50N$, can forward any packet in queue

Testing performance on lattice

Testing on *static lattice + **low** traffic*



Testing on *static lattice + **high** traffic*

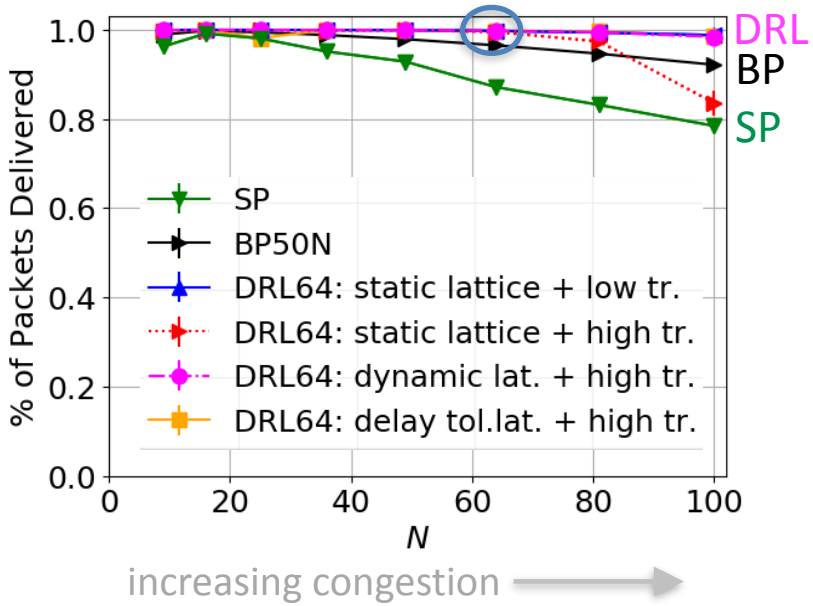


DeepRL strategies

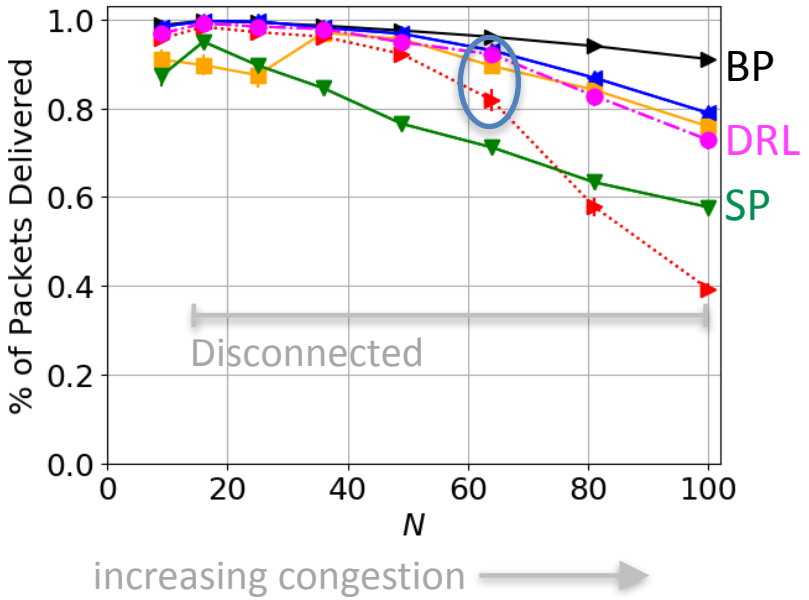
- Generalize to other values of N **and traffic levels**
- Outperform SP and BP, have **lowest delay per packet** (not shown)

Testing performance with link dynamics

Testing on ***dynamic*** *lattice + high traffic*



Testing on ***delay tol.*** *lattice + high traffic*



DeepRL strategies

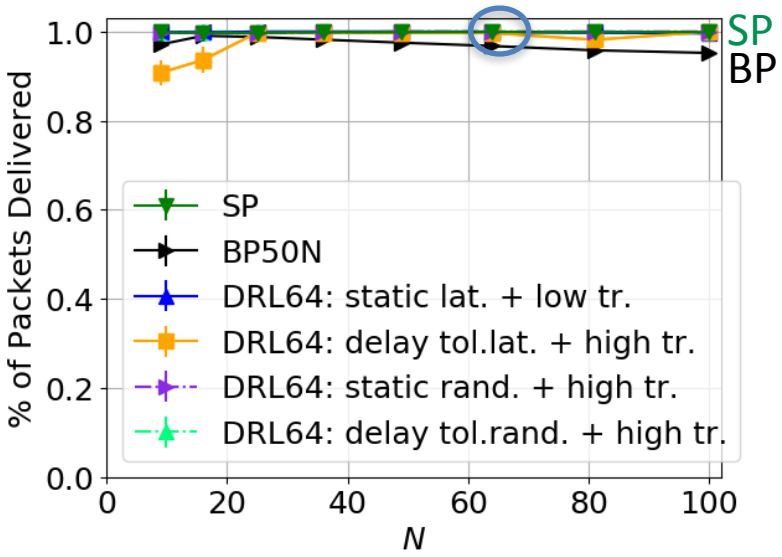
- Generalize to other values of N **and link dynamics**, have **lowest per-packet delay** (not shown)
- Outperform SP and BP except when high congestion in disconnected network

BP strategy

- Has advantage due to ability to **select any packet in queue**, along with use of **longer queues**

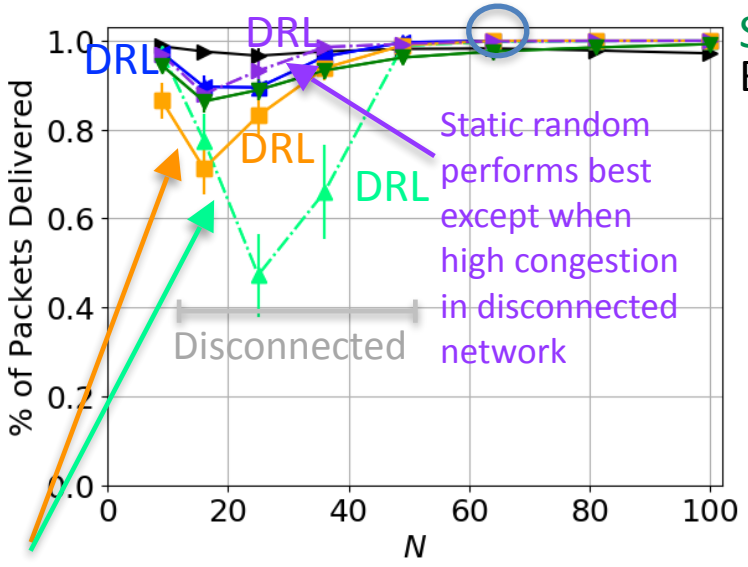
Testing performance on random

Testing on *static **random** + high traffic*



increasing congestion

Testing on *delay tol. **random** + high traffic*



increasing congestion

Delay tolerant perform best for $N = 64$ but do not generalize: need more diverse training data

Takeaways

- Possible to train offline and generalize to very different unseen scenarios
- Flexibility in choosing which packet to forward is important as network becomes disconnected/congested
- Ideally: train DeepRL agent on set of different scenarios

Outline

1. Motivation
2. Our DeepRL approach
3. Evaluation
4. **Related Work**
5. Wrap-up

Classifying RL routing approaches

	RL agent learns online	RL agent learns offline
Distributed	• Early work (table-based): [1][2] • Recent work has scalability limits, such as using network specific DNN inputs, focus on smaller networks and less congested scenarios: e.g., [3], [4]	• Recent work has scalability limits, such as using network specific DNN inputs, focus on smaller networks and less congested scenarios: e.g., [5] [6] • ***Our work:*** Our use of relational features allows scalability and generalization so that we are able to train an agent offline
Centralized (Unscalable for wireless)	• Optical transport and IP networks: [7]	• Focus is primarily SDNs, traffic engineering: e.g., [8]

[1] J. A. Boyan, M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," NIPS, 1994

[2] S. Kumar, R. Miikkulainen, "Confidence-based Q-routing: an on-line adaptive network routing algorithm," Artificial Neural Networks in Engineering, 1998.

[3] L. Chen, B. Hu, Z.-H. Guan, L. Zhao, X. Shen, "Multiagent meta-reinforcement learning for adaptive multipath routing optimization," IEEE Transactions on Neural Networks and Learning Systems, 2021.

[4] D. Mukhutdinov, A. Filchenkov, A. Shalyto, V. Vyatkin, "Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system," Future Generation Computer Systems, vol. 94, 2019.

[5] X. You, X. Li, Y. Xu, H. Feng, J. Zhao, H. Yan. "Toward packet routing with fully-distributed multi-agent deep reinforcement learning," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020.

[6] S. Kaviani, B. Ryu, E. Ahmed, K. A. Larson, A. Le, A. Yahja, J. H. Kim, "Robust and Scalable Routing with Multi-Agent Deep Reinforcement Learning for MANETs," arXiv:2101.03273, 2021.

[7] J. Suarez-Varela, A. Mestres, J. Yu, L. Kuang, H. Feng, P. Barlet-Ros, A. Cabellos-Aparicio, "Feature engineering for deep reinforcement learning based routing," ICC, 2019.

[8] A. Valadarsky, M. Schapira, D. Shahaf, A. Tamar, "Learning to route with deep RL," NIPS Deep Reinforcement Learning Symposium, 2017.

Outline

1. Motivation
2. Our DeepRL approach
3. Evaluation
4. Related Work
5. **Wrap-up**

Summary and future work

Designed novel distributed routing algorithm using relational DeepRL

Key ideas: Relational features, offline centralized training/online distributed testing, extended time action aka options

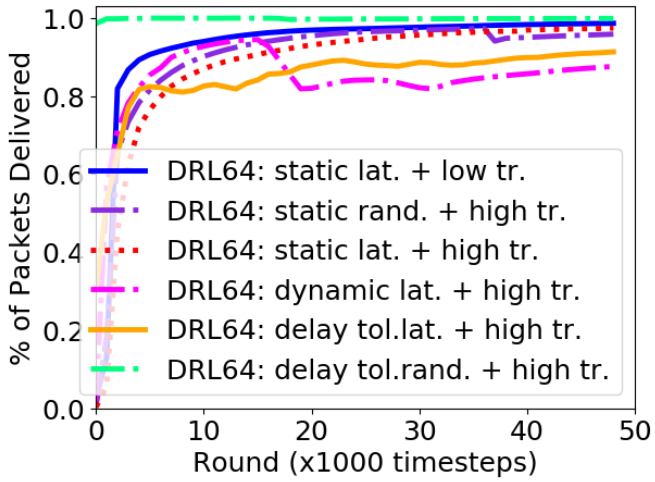
Future work:

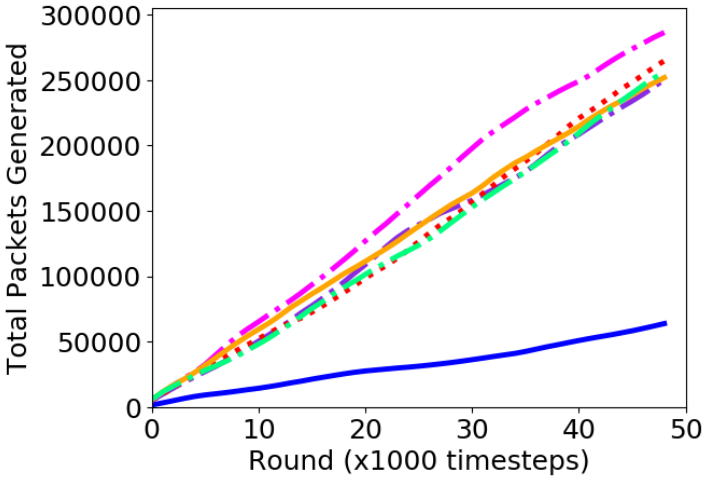
- Mobile networks
- Flexibility in which packet in queue to send
- Super DeepRL strategy trained on multiple different scenarios
- Understanding extent of generalization ability

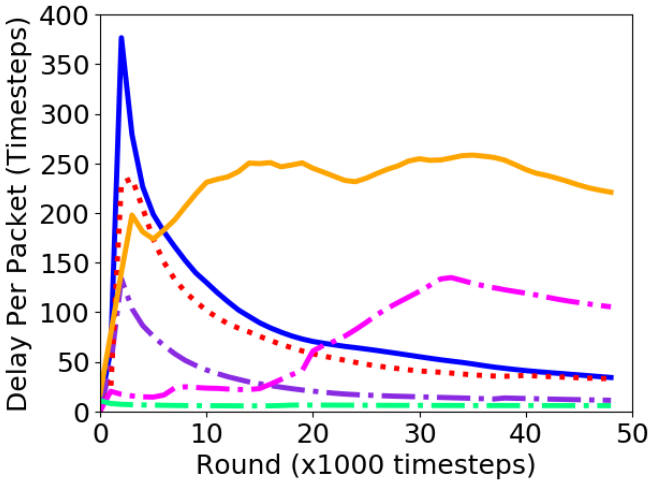
?? || /**/

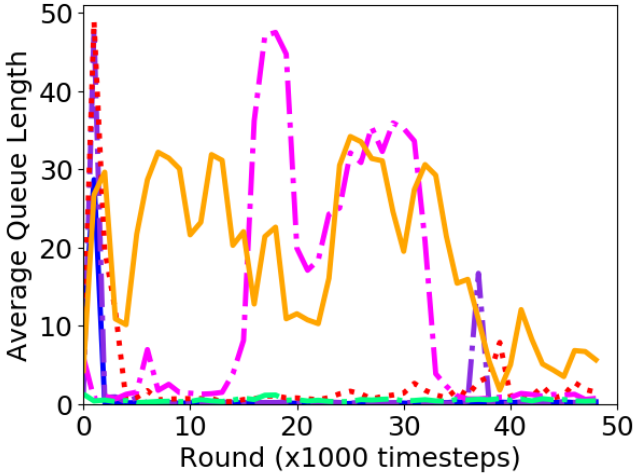
vumanfredi@wesleyan.edu
https://victoriamanfredi.com

Training performance

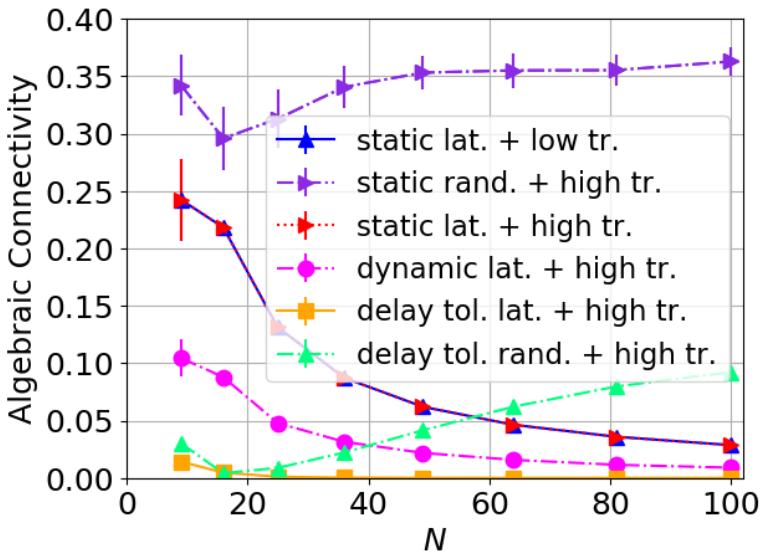


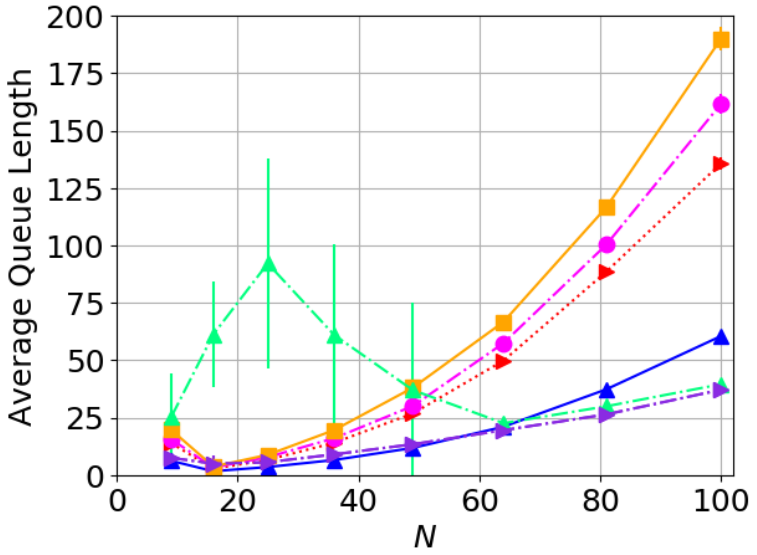


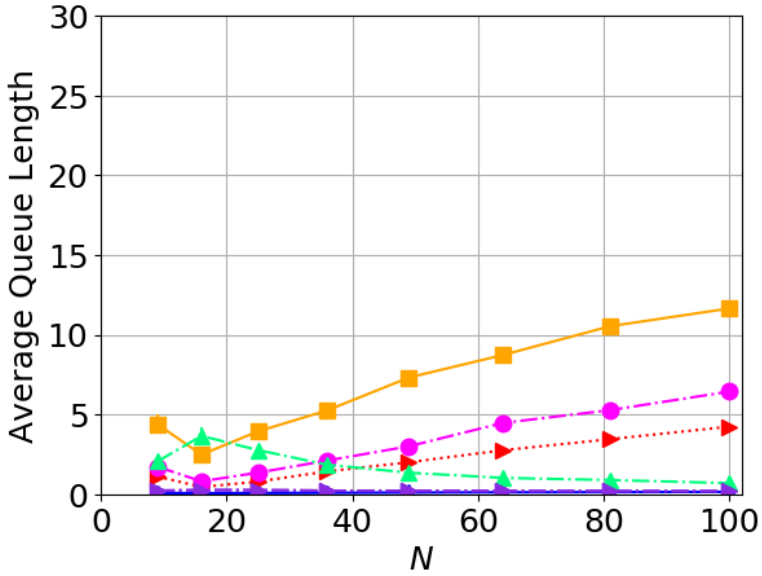


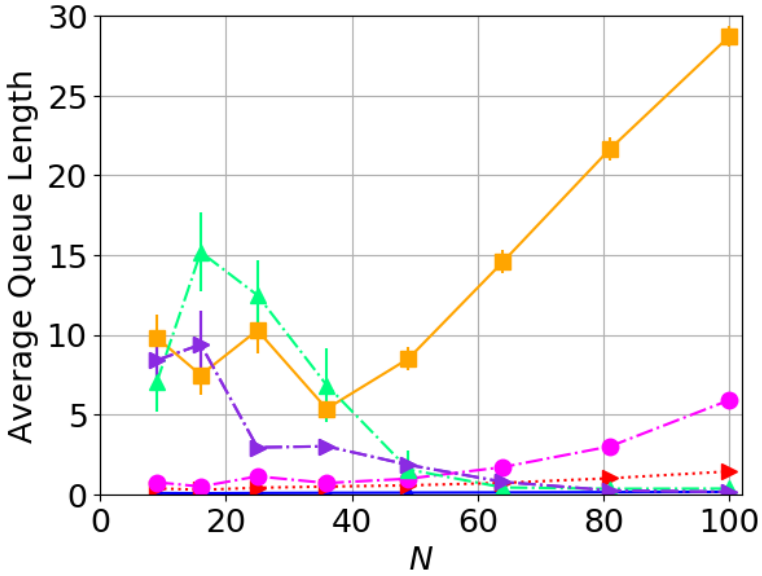


Example network connectivity and congestion





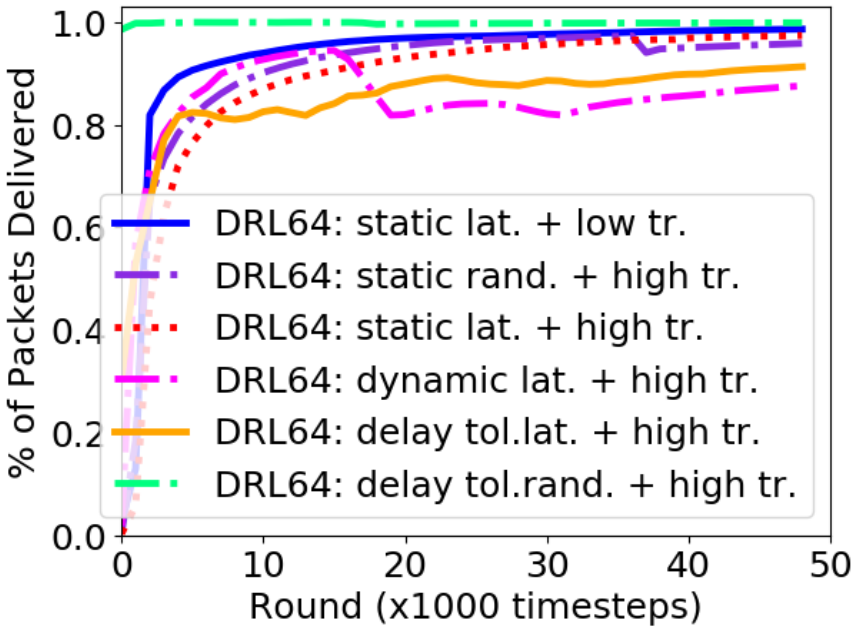


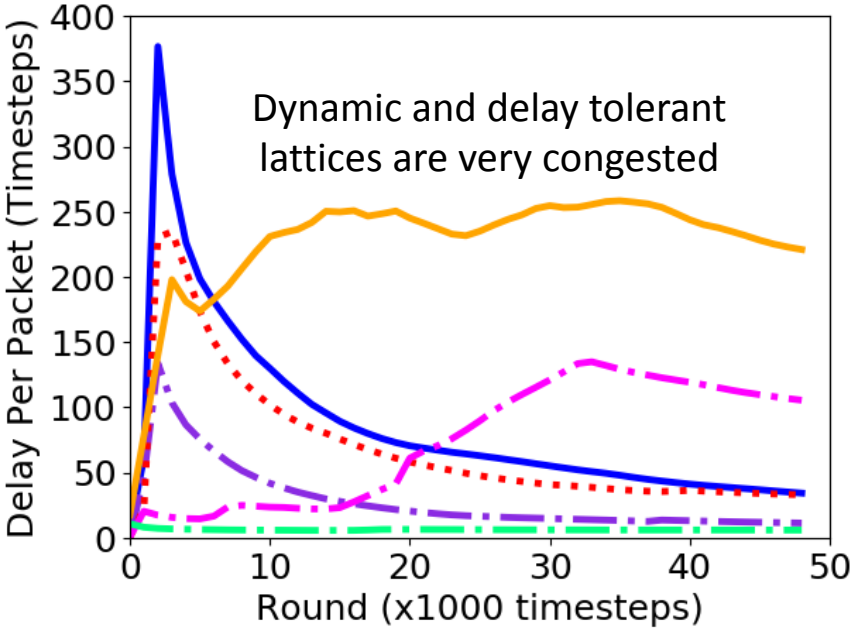


DeepRL training performance

Train **separate DeepRL agent** on each scenario for $N = 64$

- Deep RL agents converge relatively quickly to delivering most packets

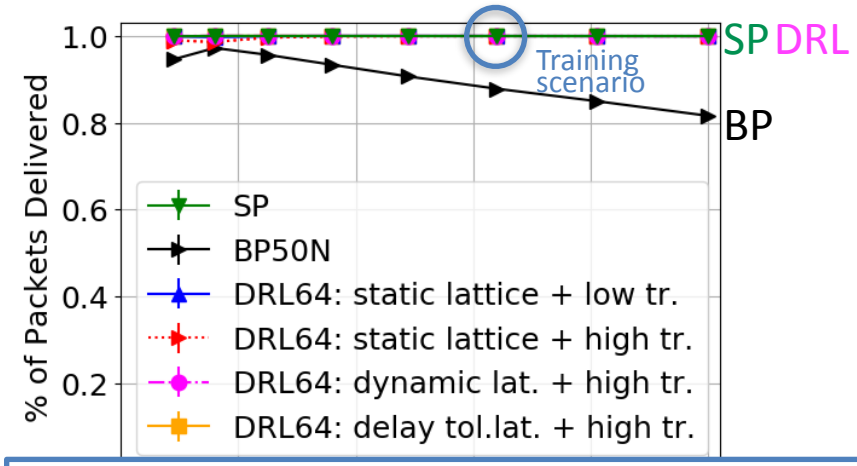




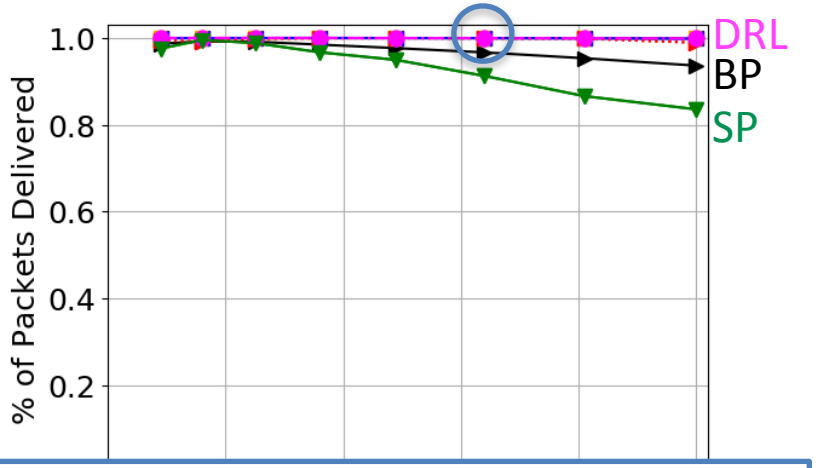
Deep RL strategies converge relatively quickly to delivering most packets

Testing performance on lattice

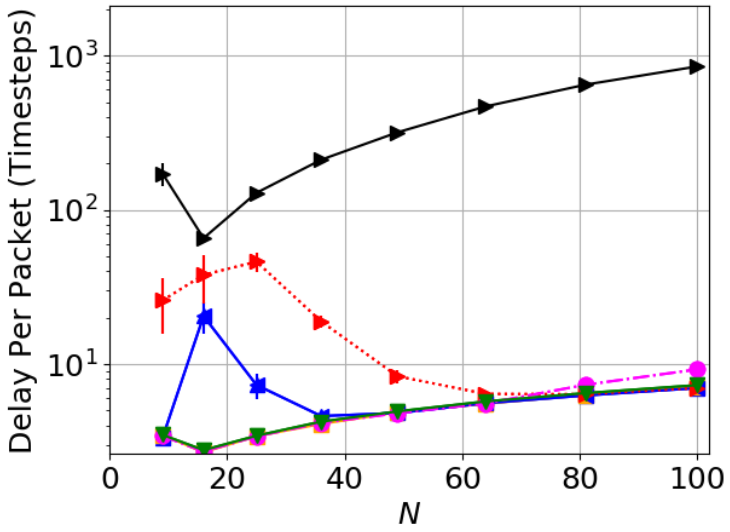
Testing on *static lattice + **low** traffic*

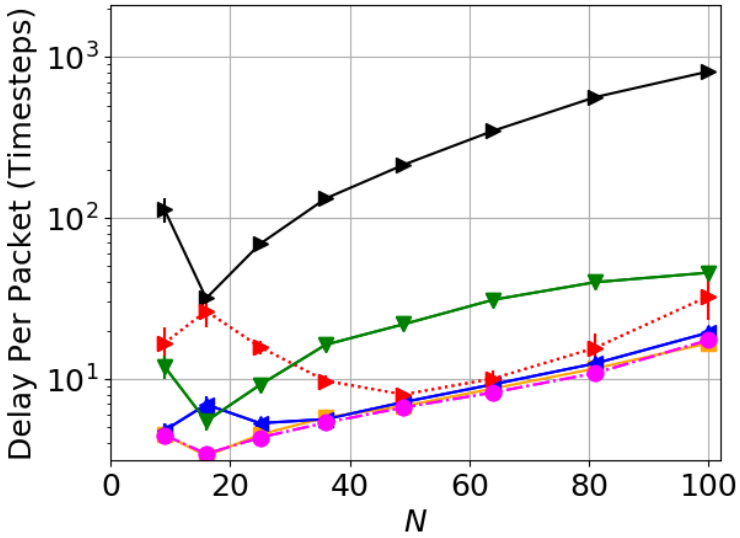


Testing on *static lattice + **high** traffic*



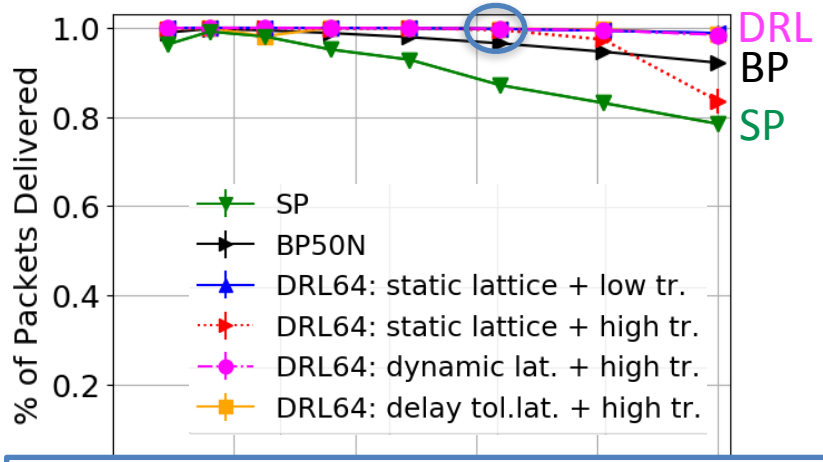
DeepRL strategies have lowest per-packet delay



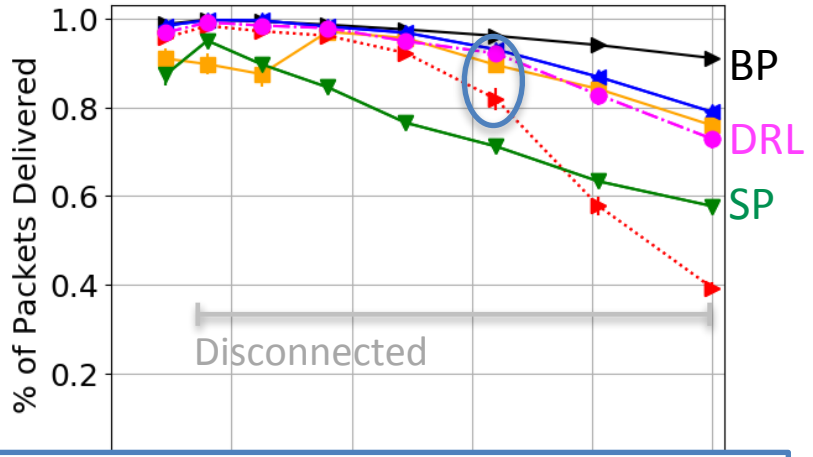


Testing performance with link dynamics

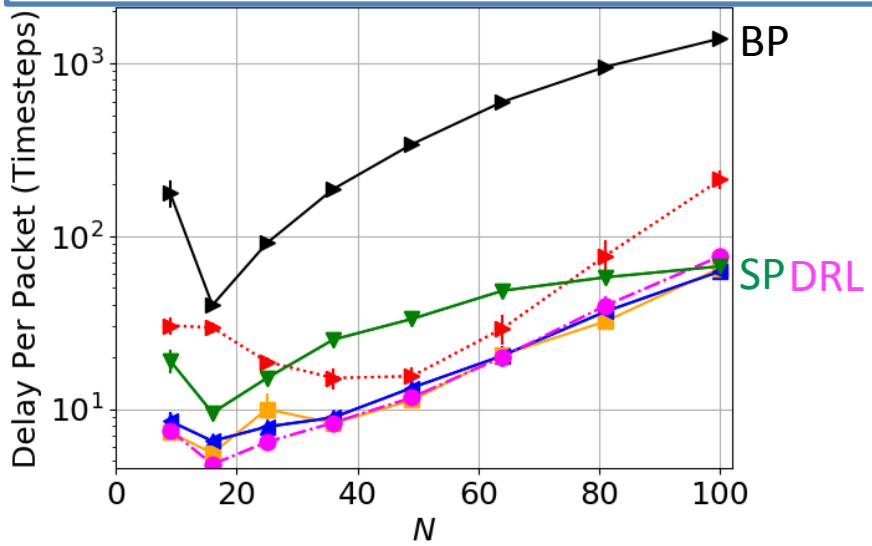
Testing on ***dynamic*** *lattice + high traffic*

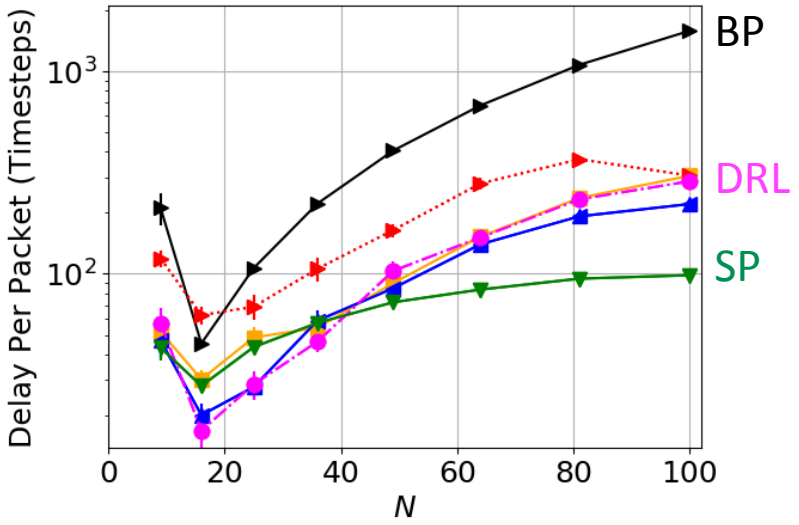


Testing on ***delay tol.*** *lattice + high traffic*



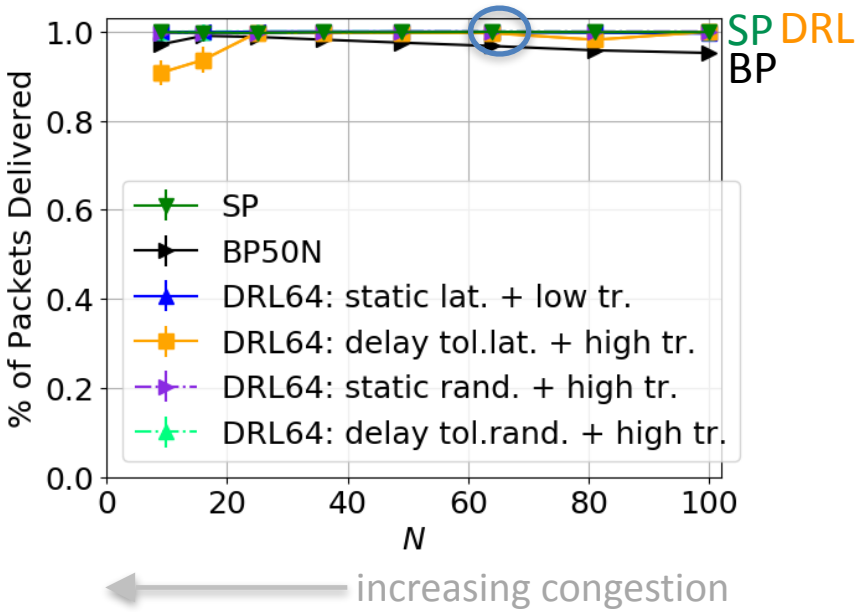
DeepRL strategy has lowest per-packet delay



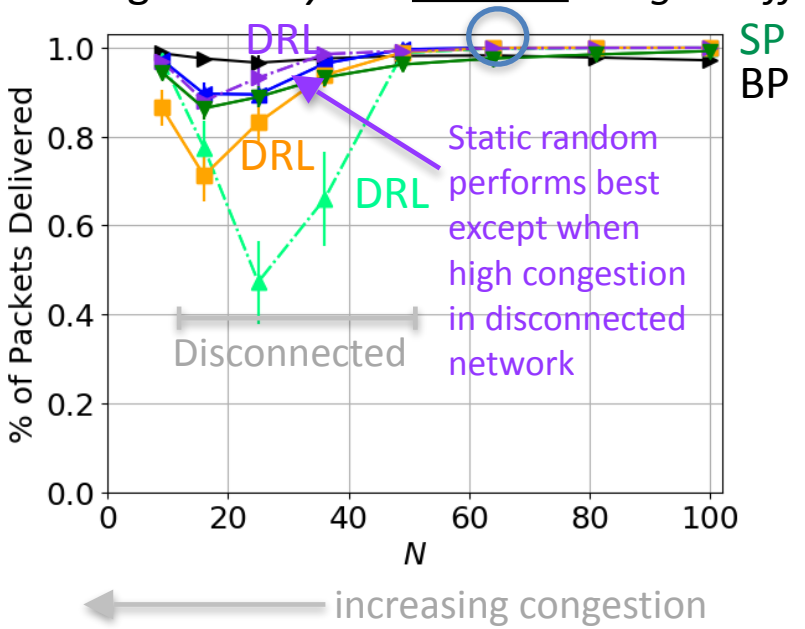


Testing performance on random

Testing on *static **random** + high traffic*



Testing on *delay tol. **random** + high traffic*



Static DeepRL strategies

- Static lattice able to generalize to highly connected and dynamic random topologies
- Static random performs best of all except when high congestion in disconnected network

Delay tolerant DeepRL strategies

- Have lowest delay for $N = 64$ and deliver all packets but do not generalize well

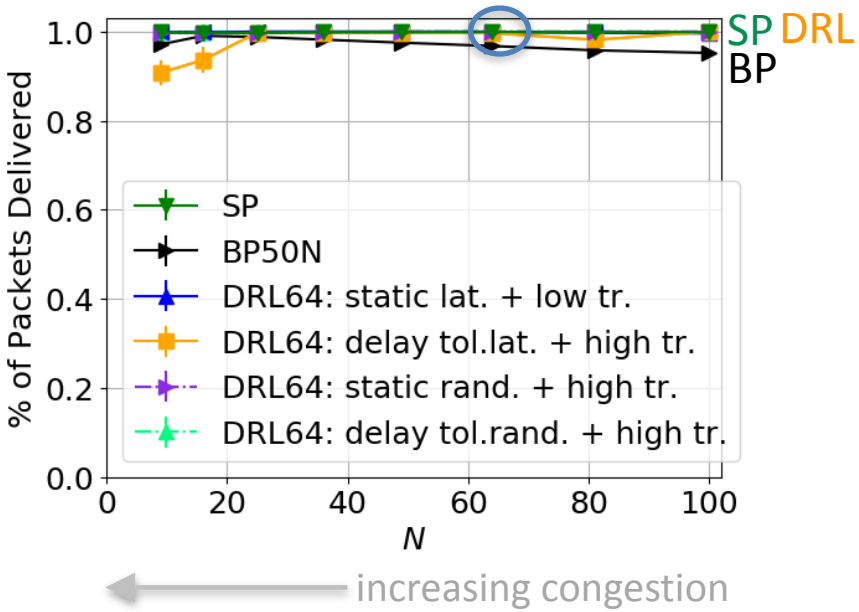
  => Need **more diversity** in training data

BP strategy

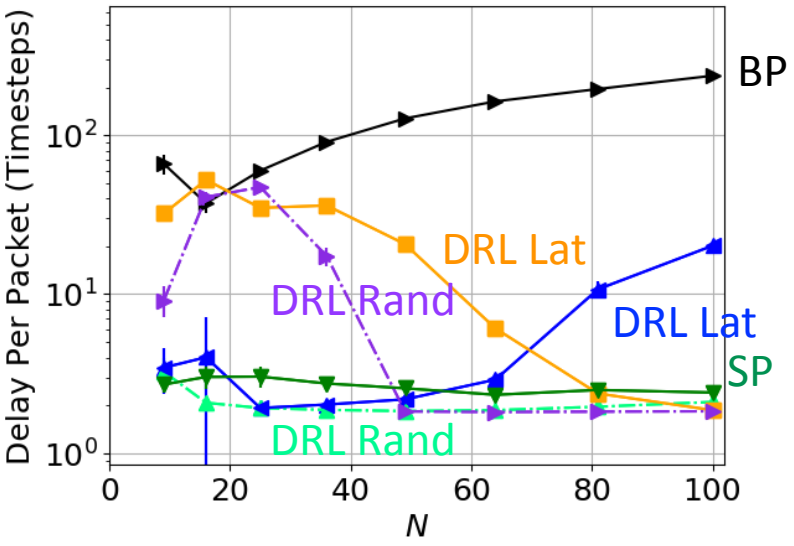
- Has advantage due to ability to **select any packet in queue**, along with use of **longer queues**

Testing performance on random

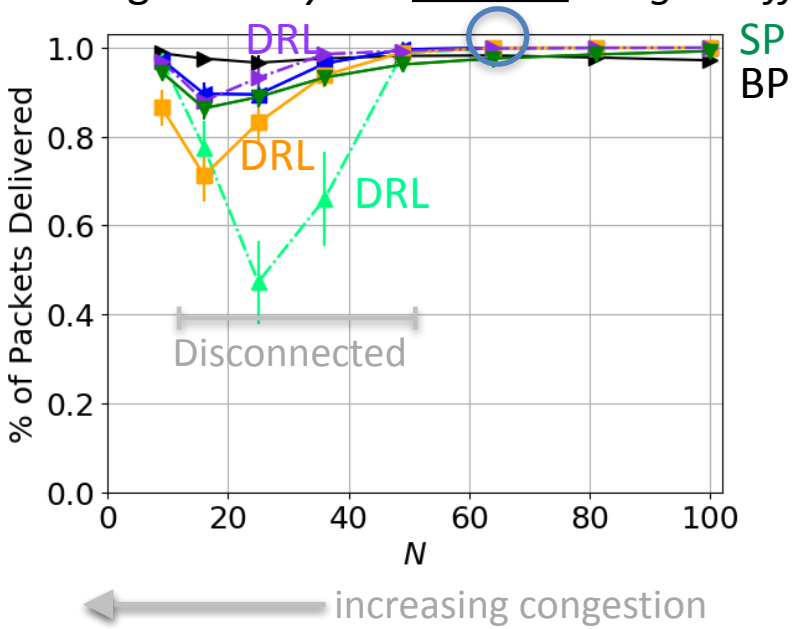
Testing on *static **random** + high traffic*



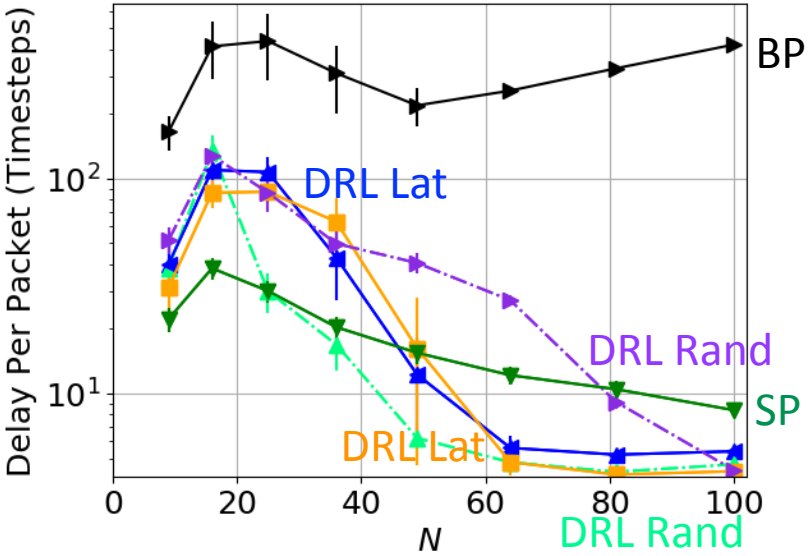
increasing congestion



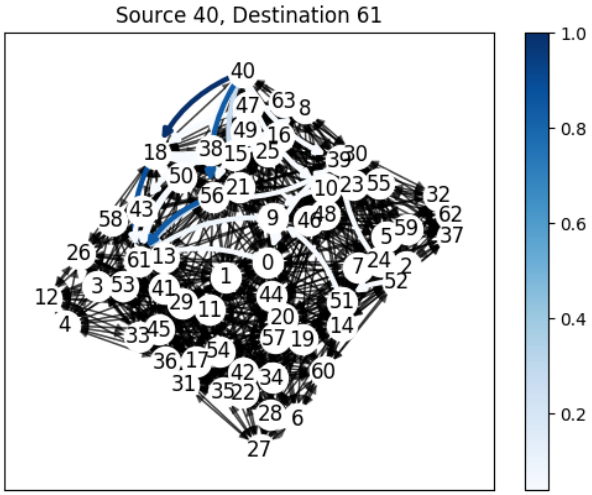
Testing on *delay tol. **random** + high traffic*



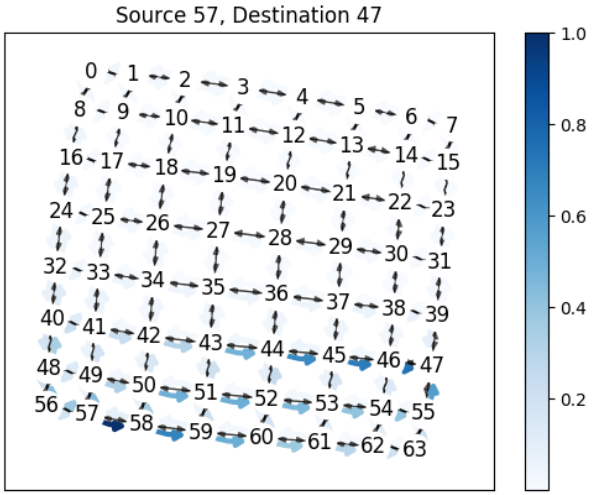
increasing congestion



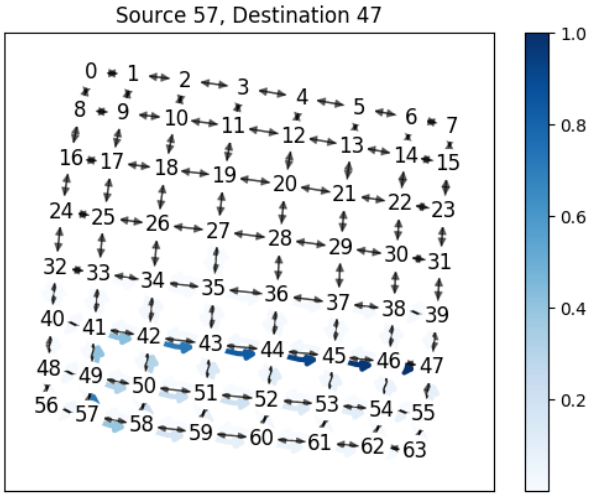
Example learned policies



Static random + high traffic



Static lattice + high traffic

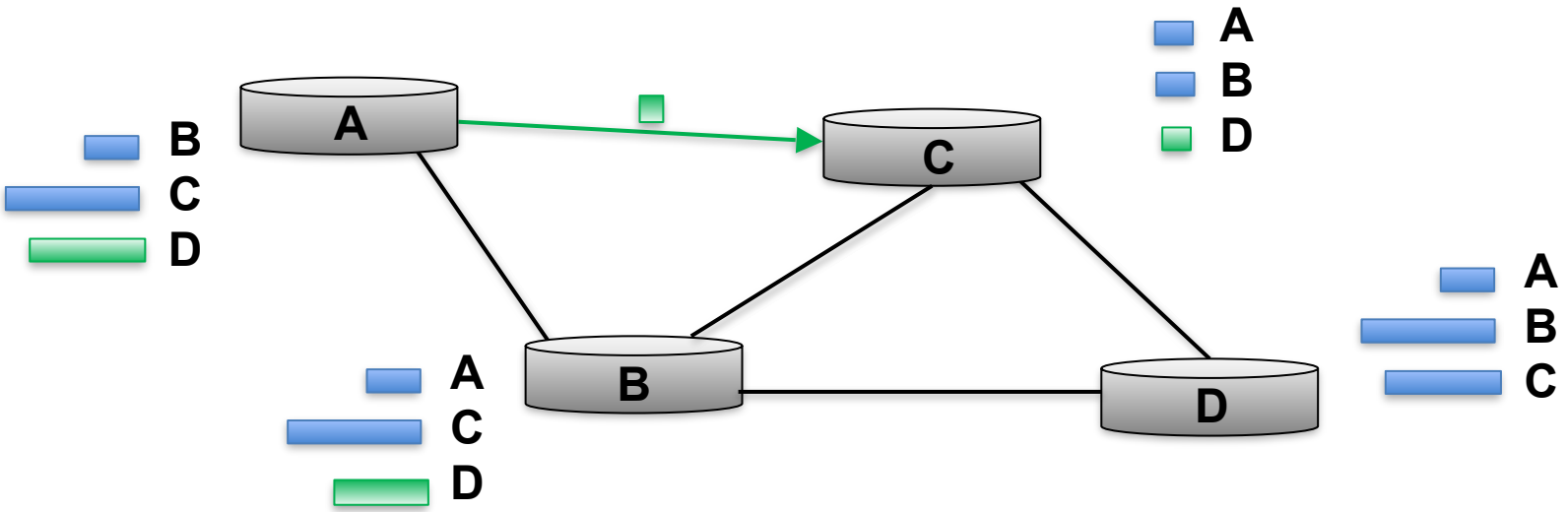


Delay tol. lattice + high traffic

Backpressure routing

Each router

- maintains queue for each destination
- uses difference in queue lengths between router and its neighbors to determine next hop for each packet



What's the problem with backpressure routing?

- high delay when insufficient traffic since queues don't fill up