Genet

Automatic Curriculum Generation for Learning Adaptation in Networking

Presenters:
Chenyue, Kuba, Sumit, Xinran

Outline

- Motivation and related works
- Curriculum learning
- Design and implementation
- Experimental evaluations

Outline

- **Motivation and related works**
- Curriculum learning
- Design and implementation
- Experimental evaluations

Introduction

- Roles of Reinforcement learning in Networking and Systems
- Roles of RL in:
  - Congestion Control (CC)
  - Adaptive Bitrate (ABR)
  - Load Balancing (LB)

Issues with RL

Issues in:

- Training on wide area Networks: Suboptimal Performance
- Training on narrow distribution: Poor Generalization

Genet

- Training framework that generate training curricula for network adaptation policies
- Built on curriculum learning
  - Gradual increase of difficulty levels of training environment

Genet as an idea

- Large gap-to-baseline -> rewarding environment
  - Difference in performance of RL policy that falls behind traditional rule-based baseline
- Genet generates RL training on environment with large gap-to-baseline RL models
- Generic in nature as it uses rule based algorithms
  - Doesn’t use handcrafted heuristics to measure the difficulty of network environment
- Genet performance as compared to RL based training:
  - ABR: 8-25% up
  - CC: 14-24% up
  - LB: 15% up

Motivation

Observing RL performance on:

- ABR: Chunk level video bitrate to the dynamics of throughput and playback buffer over the course of video session
- CC: Determines sending rate at transport layer based on sender’s observation
- LB: distributed database that reroutes each request to one of the servers

RL Performance

Use case	**Observed state** (policy input)	**Action** (policy output)	**Reward** (performance)
Adaptive Bitrate (ABR) Streaming	future chunk size, history throughput, current buffer length	bitrate selected for the next video chunk	$\sum_i(\alpha \cdot \text{Rebuf}_i + \beta \cdot \text{Bitrate}_i + \gamma \cdot \text{BitrateChange}_i)/n$
Congestion Control (CC)	RTT inflation, sending/receiving rate, avg RTT in a time window, min RTT	change of sending rate in the next time window	$\sum_i(a \cdot \text{Throughput}_i + b \cdot \text{Latency}_i + c \cdot \text{LossRate}_i)/n$
Load Balancing (LB)	past throughput, current request size, number of queued requests per server	server selection for the current request	$-\sum_i \text{Delay}_i/n$

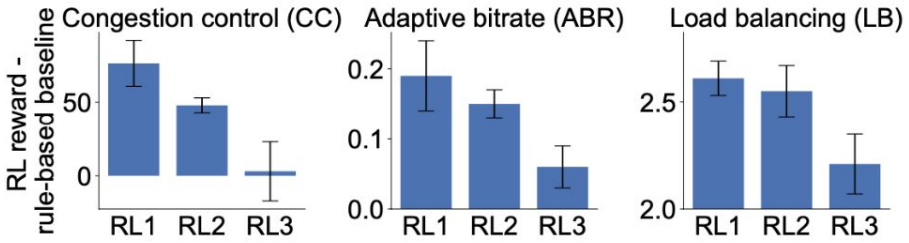
Table 1: *RL use cases in networked systems. Default reward parameters:* $\alpha = -10$ *(rebuffering in seconds),* $\beta = 1$ *(bitrate in Mbps),* $\gamma = -1$ *(bitrate change in Mbps),* $a = 120$ *(throughput in kbps),* $b = -1000$ *(latency in seconds),* $c = -2000$. *Details in A.5.*

Retrieved as Table1 from https://doi.org/10.1145/3544216.3544243

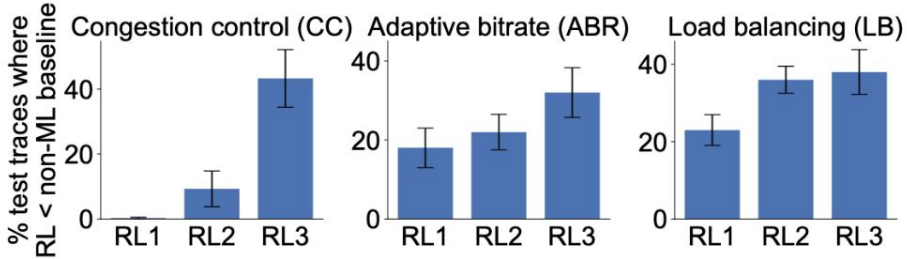
Challenge 1

Training over wide environment distributions:

- Poor performance on widespread network environments (bandwidth values)
- 3 RL distributions: RL1, RL2, RL3
- Fig(a): RL performance over baselines diminishes when range of target environment expands
- Fig(b): RL performance falling behind baselines on test environments



(a) Performance gains of RL schemes over the baselines diminish as the target distribution spans a wide range of environments.



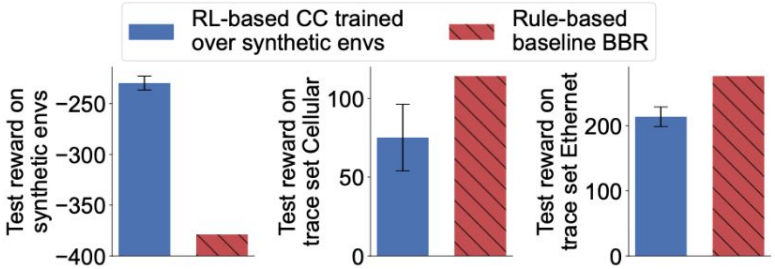
(b) Even if RL schemes perform better on average, they are worse than the baselines on a substantial fraction of test environments.

Retrieved as Figure 2 from https://doi.org/10.1145/3544216.3544243

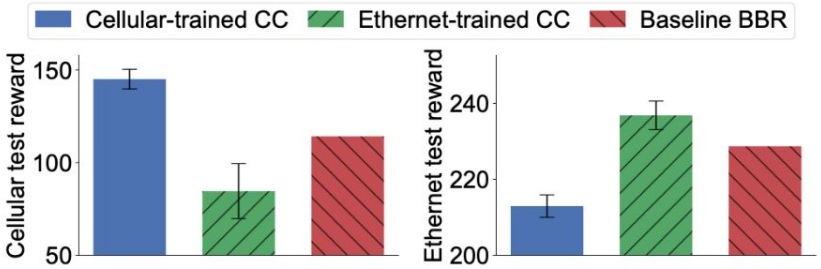
Challenge 2

Low generalizability:

- Existing RL training methods trained on one distribution not performing well on different environment distribution
- Fig(a): RL based CC trained tested on synthetic simulation, rule based baseline bbr, cellular and ethernet data from Pantheon
- Fig(b): cellular and ether trained CC tested on baseline bbr, cellular and ethernet data



(a) RL-based CC trained in synthetic network environments performs worse on real network traces than the rule-based baseline.



(b) RL-based CC trained over one real trace set performs worse on another real trace set than the rule-based baseline.

Retrieved as Figure 3 from https://doi.org/10.1145/3544216.3544243

Outline

- Motivation and related works
- **Curriculum learning**
- Design and implementation
- Experimental evaluations

Big Goal

- How to improve RL training such that the learned adaptation policies achieve good asymptotic performance **across a broad range of target network environments**?

Curriculum learning in RL

- Unlike the traditional RL training that samples training environments from a fixed distribution in each iteration, curriculum learning **varies the training environment distribution to gradually increase the difficulty of training environments**
- Training will see more environments that are more likely to improve, which we refer to as **rewarding environments**
- In many RL applications, prior work has shown the promise of curriculum learning, namely **faster convergence, higher asymptotic performance, and better generalization**

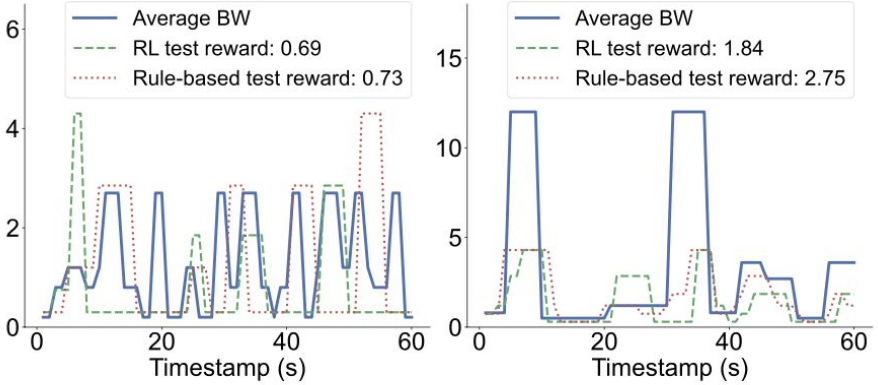
Curriculum Learning Theory

- Curriculum allows the model to optimize a family of **gradually less smooth loss functions** and **prevents it from being trapped in local minima**
- In the early stage of the curriculum, easier training samples are selected to **comprise a smoothed loss function that reveals the big picture and is easier to optimize**. The resulting model serves as a good starting point when more difficult samples are introduced to the training, **reducing the smoothness of the loss function and making it harder to optimize**.

Big question: how to define what is a rewarding environment? Here are three strawman approaches

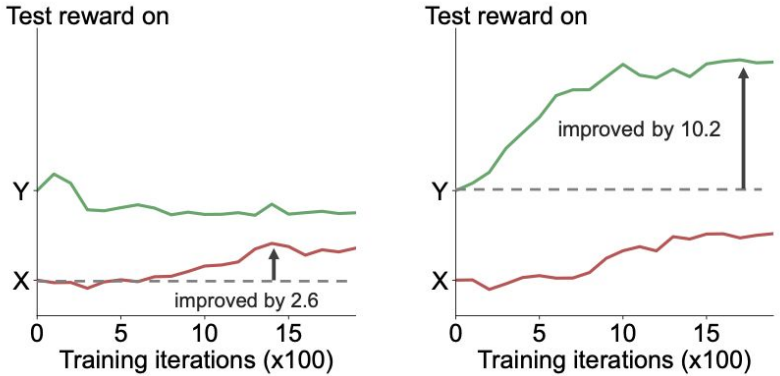
- Inherent properties
  - E.g. in congestion control, network traces with higher bandwidth variance
  - This only differs environment in handpicked properties but not something more complex
- Performance of rule based methods
  - While it can distinguish any two environments, it does not hint at how to improve RL
- Performance gap to the optimum
  - Optimum is hard to get

Discussion: could you infer from the graphs here why using gap to optimum approach fails here?



(a) Trace in X (hard)

(b) Trace in Y (improvable)



(a) Add X to training

(b) Add Y to training

Outline

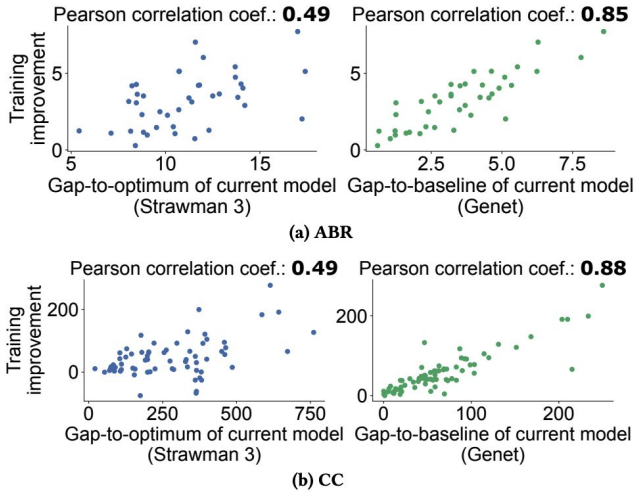
- Motivation and related works
- Curriculum learning
- **Design and implementation**
- Experimental evaluations

Curriculum generation

- *Gap-to-baseline*: Genet identifies rewarding environments where RL policy performs worse than rule-based baselines by a large margin
  - RL policy could learn to imitate known rules and perform at least as good as baselines.
  - While RL trained policy is sensitive to what environments are seen during training, traditional rule-based baselines are less susceptible to such discrepancies
  - So rule-based methods are seen as complementary to RL-trained models

Curriculum generation

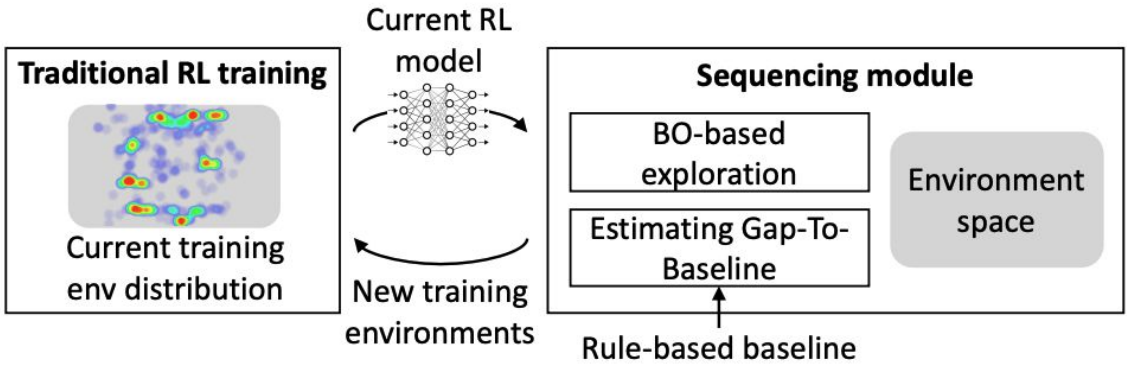
- Compare gap-to-baseline and gap-to-optimum
  - There are more potential room to improve current RL model by using gap-to-baseline than gap-to-optimum as metrics in ABR and CC environments



Left: amount of training improvements w.r.t gap-to-optimum values.
Right: amount of training improvements w.r.t gap-to-baseline values.

Training framework: overview

- Repeat:
  - Update RL model over current training environment distribution for some number of iterations
  - Use gap-to-baseline metrics to select rewarding environments compared to the current model
  - Change the next training environment distribution by promoting these rewarding environments



Training framework: traditional RL training

- Environment configurations in synthetic simulators
  - E.g. bandwidth 2-3 Mbps, bandwidth changing frequency 0-20s, buffer length 5-10s
- Augment training environments with real traces when available
  - E.g. categorize bandwidth trace with related parameters; sample a bandwidth trace whose parameters fall into the range of selected configurations
- When training the policy, training environments are uniformed sampled from distribution of configurations

Training framework: sequencing module

- After fixed number of iterations, current RL model and a pre-determined rule-based baseline are used to search for rewarding environments with high gap-to-baseline values
  - Ideally want to evaluate on all possible inputs in environment configuration space
- Approach: Bayesian optimization (BO) approximates function value Gap(p) given input p from configuration space $Gap(p) = R(\pi^{rule}, p) - R(\pi_\theta^{rl}, p)$
  - Evaluate in k=10 randomly generated environments from configuration p to perform search
  - Update environment distribution for next iterations of training: $\Omega_{cur} \leftarrow (1 - w) \cdot \Omega_{cur} + w \cdot \{p_{new}\}$

Design choices and implementations

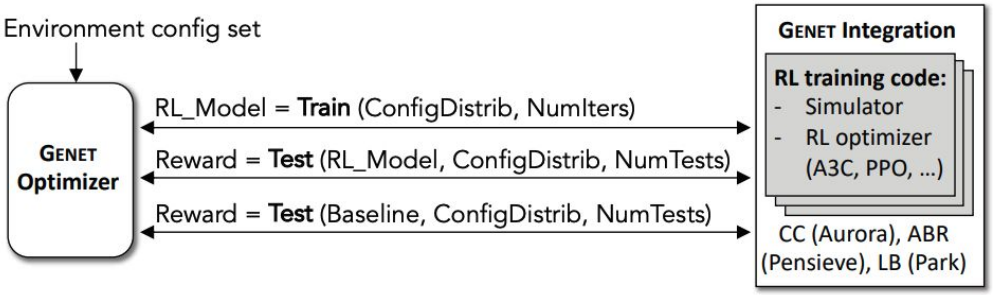
- How to choose rule-based baselines?
  - Don’t have to be optimal and shouldn’t fail under simple cases
    - E.g. MPC in ABR; Cubic and BBR in CC; shortest-job-first in LB
  - Potential improvement: ensemble of rule-based baselines as heuristics. Prioritize environments where RL policy worse than anyone in the set of baselines
- Discussion: What if a rule-based baselines does not exist?
  - Reduce to traditional RL training but lose benefits of curriculum learning
  - Use gap between optimal solution based on ground truth knowledge and current RL model
  - Treat a trained RL model as a rule-based baseline
  - Other possibilities?

Design choices and implementations

- Why to use BO?
  - Challenges: 1) high-dimensional search space, 2) computationally expensive to evaluate a gap-to-baseline value
  - Potential alternative: set a threshold for gap-to-baseline value under selected environment
    - Searching this way may not end if current RL model already better than baseline
    - Introduce additional complexity to tune this hyperparameter
- How to deal with forgetting?
  - Ideally we want to train on whole space of environment configurations
  - Model is still trained on environments from initial distribution until the end, since newly updated distribution is a weighted combination of original and identified rewarding environments

Design choices and implementations

- Integrate with existing algorithms: Pensieve for ABR, Aurora for CC, Park for LB



Outline

- Motivation and related works
- Curriculum learning
- Design and implementation
- **Experimental evaluations**

Evaluation Setup - Overview

1. Synthetic environments
2. Trace-drive environments
3. Baselines comparison
    a. Genet vs. traditional-RL algorithms
    b. Genet vs. traditional rule-based algorithms

Evaluation Setup - Overview

1. Synthetic environments
2. Trace-drive environments
3. Baselines comparison
    a. Genet vs. traditional-RL algorithms
    b. Genet vs. traditional rule-based algorithms

We show:

- Genet improves the *performance* and *generalization* of RL algorithms
- Genet policies have a higher chance to outperform rule-based baselines
- Genet's design choices are effective against alternatives

Evaluation Setup

1. Synthetic environments
    a. Using §A.2 and Tables 3, 4, 5
    b. Chosen parameters impacts RL performance
2. Trace-drive environments
    a. Real traces for CC and ABR across training/testing environments
    b. ABR tested w/ pre-recorded video over 290 traces from FCC broadband measurements
    c. CC tested w/ 121 cellular traces and 112 Ethernet traces

Name	Use case	Training # traces, total length (s)	Testing # traces, total length (s)
FCC	ABR	85, 105.8k	290, 89.9k
Norway	ABR	115, 30.5k	310, 96.1k
Ethernet	CC	64, 1.92k	112, 3.35k
Cellular	CC	136, 4.08k	121, 3.64k

Table 2: *Network traces used in ABR and CC tests.*

ABR Parameter	RL1	RL2	RL3	Default	Original
Max playback buffer (s)	[2, 10]	[2, 50]	[2, 100]	60	60
Video chunk length (s)	[1, 4]	[1, 6]	[1, 10]	4	4
Min link RTT (ms)	[20, 30]	[20, 220]	[20, 1000]	80	80
Video length (s)	[40, 45]	[40, 200]	[40, 400]	196	196
Bandwidth change interval (s)	[2, 2]	[2, 20]	[2, 100]	5	
Max link bandwidth (Mbps)	[2, 5]	[2, 100]	[2, 1000]	5	

Table 3: *Parameters in ABR simulation. Colored rows show the configurations (and their ranges) used in the simulator in the original paper. The synthetic trace generator is described in §A.2.*

CC Parameter	RL1	RL2	RL3	Default	Original
Maximum link bandwidth (Mbps)	[0.5, 7]	[0.4, 14]	[0.1, 100]	3.16	[1.2, 6]
Minimum link RTT (ms)	[205, 250]	[156, 288]	[10, 400]	100	[100, 500]
Bandwidth change interval (s)	[11, 13]	[8, 3]	[0, 30]	7.5	
Random loss rate	[0.01, 0.014]	[0.007, 0.02]	[0, 0.05]	0	[0, 0.05]
Queue (packets)	[2, 6]	[2, 11]	[2, 200]	10	[2, 2981]

Table 4: *Parameters in CC simulation. Colored rows show the configurations (and their ranges) used in the simulator in the original paper. The synthetic trace generator is described in §A.2. The range of RL1 is defined as 1/9 of the range of RL3 and the range of RL2 is defined as 1/3 of RL3. The CC parameters shown here for RL1 and RL2 are example sets.*

LB Parameter	RL1	RL2	RL3	Default	Original
Service rate	[0.1, 2]	[0.1, 5]	[0.1, 10]	[0.5, 1.0, 2.0]	[2, 4]
Job size (byte)	[100, 200]	[100, 10^3]	[1, 10^4]	2000	[100, 1000]
Job interval (ms)	[0.01, 0.05]	[0.01, 0.1]	[0.1, 1]	0.1	0.2
Number of jobs	[10, 100]	[10, 1000]	[10, 5000]	2000	1000
Queue shuffled probability	[0.1, 0.2]	[0.1, 0.5]	[0.1, 1]	0.5	

Table 5: *Parameters in LB simulation. Colored rows show the configurations (and their ranges) used in the simulator in the original paper. The synthetic trace generator is described in §A.2.*

Evaluation Setup

3. Baselines comparison

- Compare against *traditional RL-trained policies* (“baseline” ML)
  - RL1, RL2, RL3 from previous slide
  - Compare against trace-driven environments

- Compare against *traditional rule-based algorithms* (“baseline” non-ML)
  - ABR (adaptive bitrate streaming) → BBA, RobustMPC
  - CC (congestion control) → PCC-Vivance, BBR, Cubic
  - LB (load-balancing) → Least-load-first

Evaluation - Asymptotic Performance

Compare *Genet-trained policies* vs. *traditionally-trained RL policies*

1. Synthetic environments

2. Trace-driven environments

Evaluation - Asymptotic Performance

Synthetic environments

- Traditional-RL training shows little improvement over rule-based baselines
- Genet-trained CC, ABR, LB policies do:
  - (Under RL3 synthetic range)
  - 8–25% for ABR
  - 14– 24% for CC
  - 15% for LB

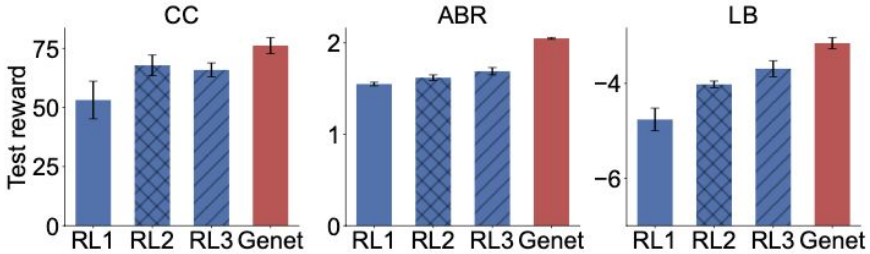


Figure 9: *Comparing the performance of GENET-trained RL policies for CC, ABR, and LB, with baselines in unseen synthetic environments drawn from the training distribution, which sets all environment parameters to their full ranges.*

Evaluation - Asymptotic Performance

Synthetic environments in example scenario (ABR)

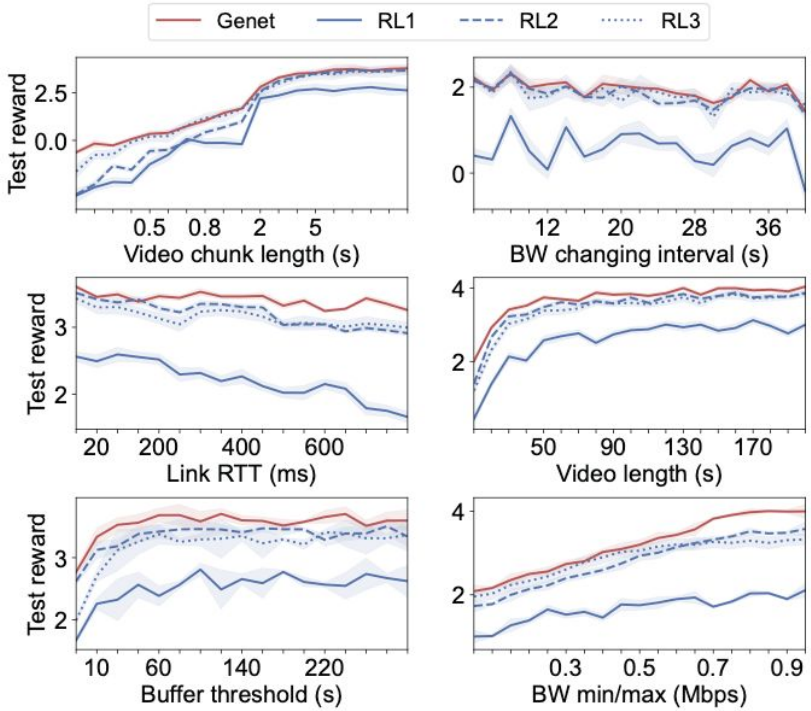


Figure 10: *Test of ABR policies along individual env-parameters.*

Evaluation - Asymptotic Performance

Synthetic environments in example scenario (LB)

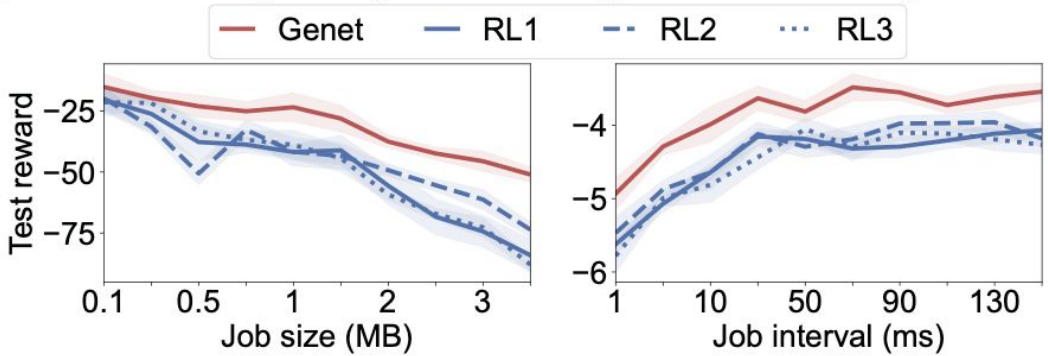
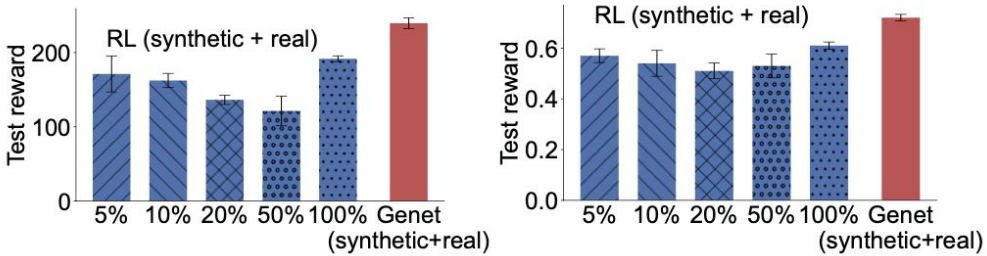


Figure 11: *Test of LB policies along individual env-parameters.*

Evaluation - Asymptotic Performance

Trace-driven environments

- Tested with genet-trained CC, ABR algorithm
  - Using FCC & Norway for ABR training/testing
  - Using Ethernet & Cellular for CC training/testing
- Uses a combination of real-to-synthetic traces for training



(a) Congestion control (CC) (b) Adaptive bitrate (ABR)

Figure 12: *Asymptotic performance of GENET-trained CC policies (a) and ABR policies (b) and baselines, when the real network traces are randomly split into a training set and a test set.*

Evaluation - Generalization to trace-driven environments

Genet and *traditional-RL* policies for ABR and CC

- trained over synthetic environments (RL3 range)
- tested on trace-driven environments from previous table

Extreme case of previous study:

When no real data is available during training, Genet still outperforms the traditional-RL models on trace-driven environments!

Evaluation - Generalization to trace-driven environments

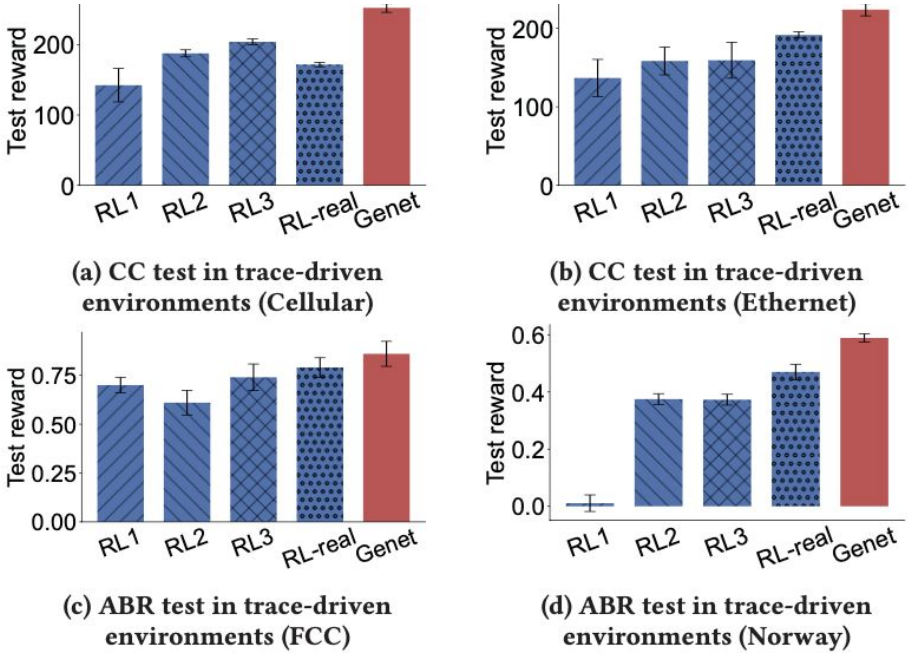
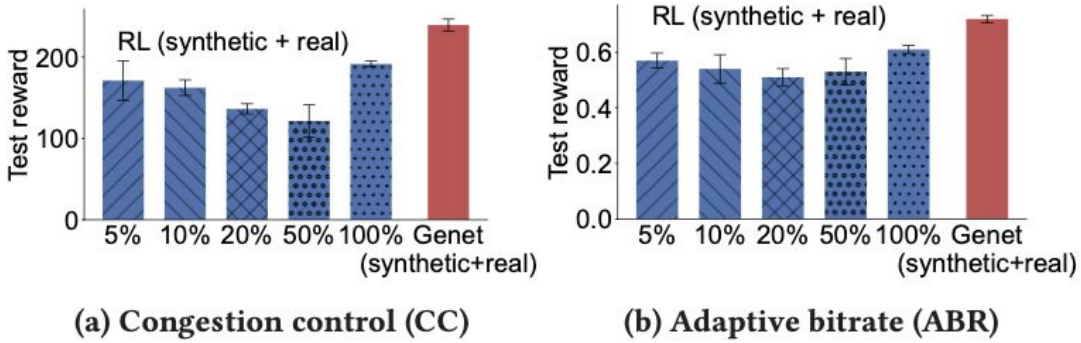


Figure 13: *Generalization test: Training of various methods is done entirely in synthetic environments, but the testing is over various real network trace sets.*

Evaluation - Generalization to trace-driven environments

- Asymptotic performance when traces are used to train



(a) Congestion control (CC) (b) Adaptive bitrate (ABR)

Figure 12: *Asymptotic performance of GENET-trained CC policies (a) and ABR policies (b) and baselines, when the real network traces are randomly split into a training set and a test set.*

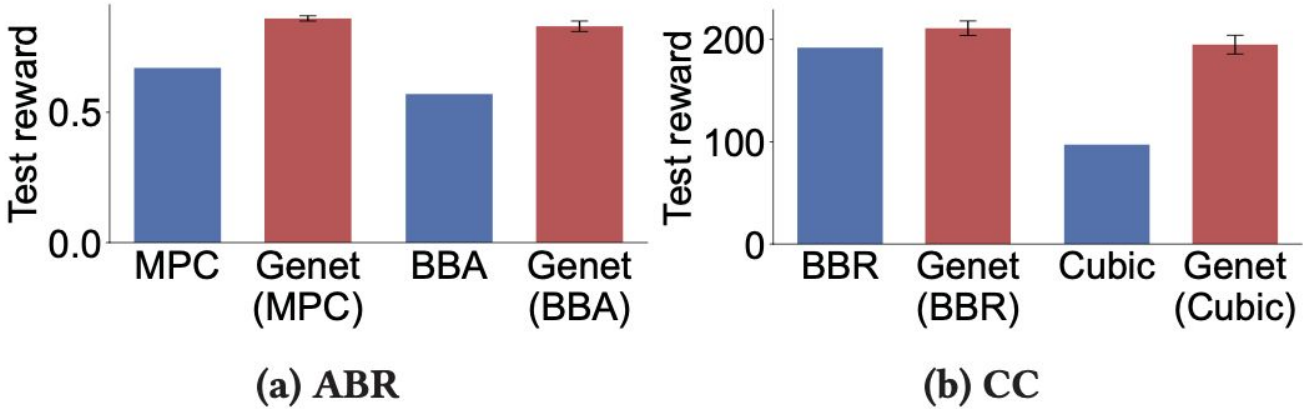
Evaluation against rule-based baselines

- Choosing meaningful rule-based baselines?
- How likely is Genet to outperform these rule-based baselines?
- Again, how does this playout with real-world tests?
- Why Genet vs. other design choices?

Using naive/bad baselines

- Want to prove that we should use relatively good baselines
  - Authors used bad baseline: choosing highest bitrate when rebuffering in ABR and choosing highest load server in load balancing
  - In each case, BO based search fail to find good training environment b/c RL quickly outperforms baseline
  - Suggests that negative impact of using naive baseline is restricted to selecting environments

Performance compared to rule based methods in synthetic environments



Performance in real environment

- The author also test the Genet-trained ABR and CC policies in five real wide-area network paths and track real network trafficking .
- For statistical confidence, the authors run the Genet-trained policies and their baselines back-to-back, each at least five times
- Outperforms in all but two case

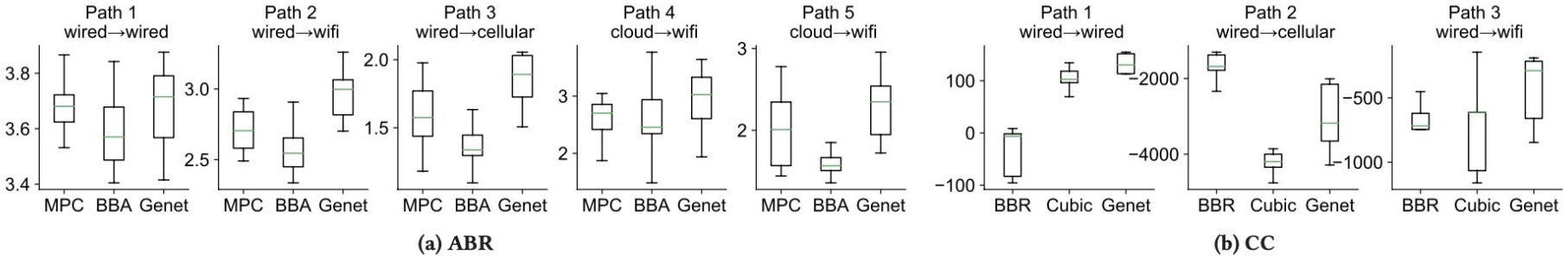
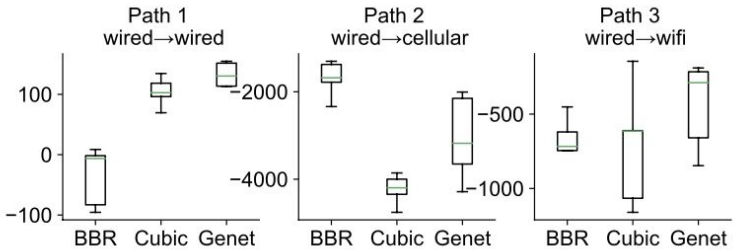


Figure 16: *Testing ABR and CC policies in real-world environments.*

Failure case

- Negative performance in path 2 in CC because network has a deeper queue than in training
- Show that policy does not generalize to all environment but only for certain range



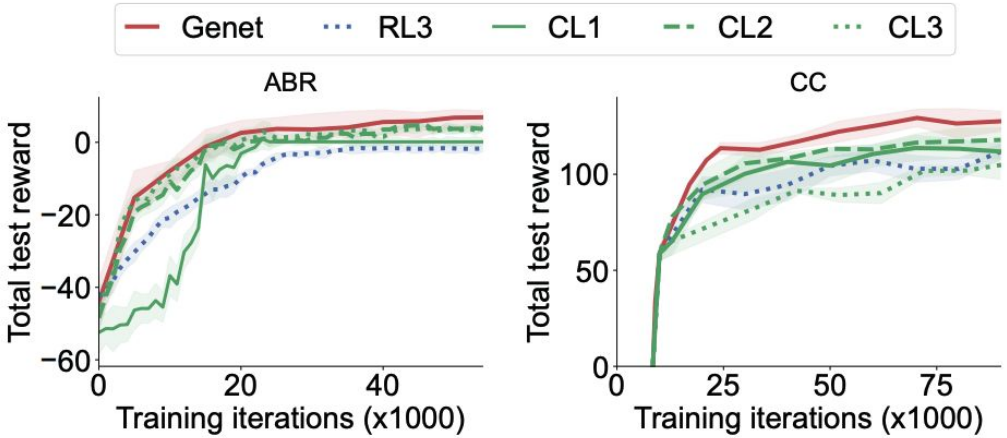
(b) CC

Path	CC	Throughput (Mbps)	90th percentile latency (ms)	Packet loss rate	Reward
Path 1	BBR	164.2	57.25	0.0906	-35.62
	Cubic	158.2	56.60	0.0072	104.2
	GENET	180.5	55.54	0.0063	152.1
Path 2	BBR	0.2108	3346	0.0407	-1721
	Cubic	0.2149	6978	0.2206	-4273
	GENET	0.1975	6381	0.0267	-3178
Path 3	BBR	5.40	1581	0.0136	-705.9
	Cubic	6.63	1400	0.0382	-719.1
	GENET	4.91	1180	0.0075	-439.9

Table 7: *Reward breakdown of Figure 16(b) in CC real-world experiments.*

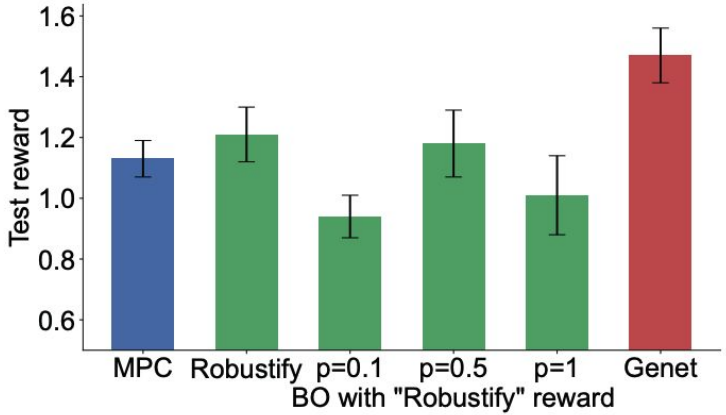
Effects of design choices

- Compare to other curriculum learning schemes:
  - CL1: hand-picked heuristics by increasing bandwidth fluctuation frequency
  - CL2: gradually picking environments where rule-based baselines perform bad
  - CL3: adding traces using gap-to-optimum metrics



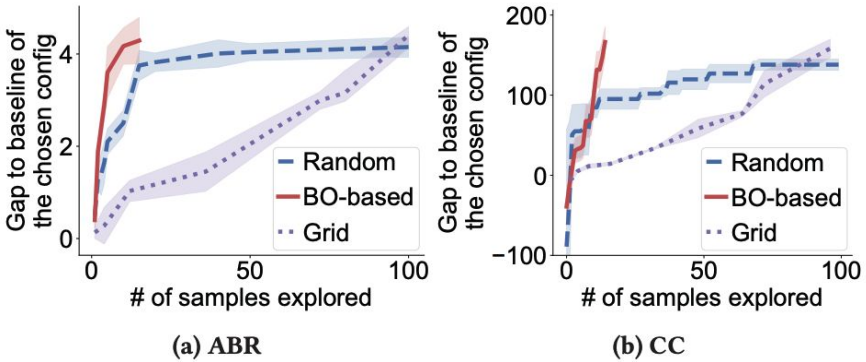
Effects of design choices

- Compare to “Robustifying” baseline:
  - Robustifying baseline: train another policy to improve current model by generating adversarial network traces in ABR environments
  - BO w/ Robustifying reward: search for configurations that maximize the gap between current model and optimal reward, penalized by bandwidth non-smoothness at different weights



Effects of design choices

- Compare to efficiency of other search approaches:
  - Random search: uniform sample from environment configuration space
  - Grid search: search and update each configuration one by one
- BO searches for much fewer steps and able to find high gap-to-baseline values compared to other baselines



Thanks!