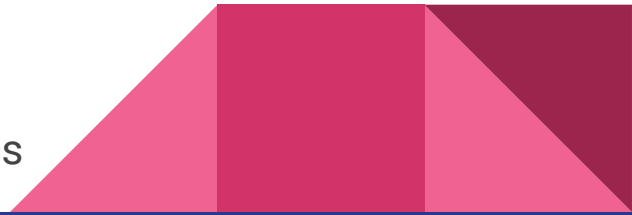
Park: An Open Platform for Learning-Augmented Computer Systems

Hongzi Mao, Parimarjan Negi, Akshay Narayan, Hanrui Wang, Jiacheng Yang, Haonan Wang, Ryan Marcus. Ravichandra Addanki, Mehrdad Khani, Songtao He, Vikram Nathan, Frank Cangialosi, Shaileshh Bojja Venkatakrishnan, Wei-Hung Weng, Song Han, Tim Kraska, Mohammad Alizadeh

Presented by Claire Coffey for R244

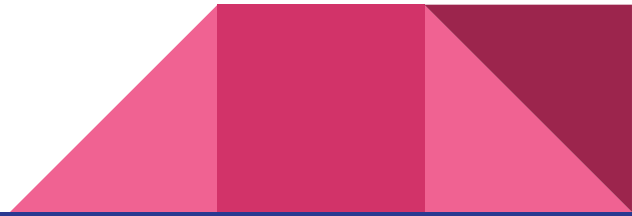
Background

- Reinforcement learning powerful for sequential decision making problems
  - Game playing etc, mostly controlled environments
- Seldom used in real-world or systems applications
  - But potential for success in systems
- Systems present new challenges for RL
  - Systems are complex - vast landscape of decision-making problems
  - E.g. dynamically varying jobs and machines in clusters, data-flow graphs, highly stochastic environments
  - Difficult to model accurately
  - Human solutions often suboptimal
  - Good case for using RL
  - Low cost of experimentation compared to e.g. robotics



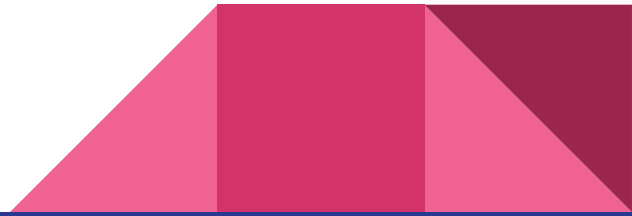
Motivation

- Some recent work to address challenges in RL for systems
  - E.g. Mao et al. variance reduction for reinforcement learning in input-driven environments
- Lack of good benchmarks for evaluating solutions of RL in systems
- Absence of a simple platform for experimenting with RL algorithms in systems problems



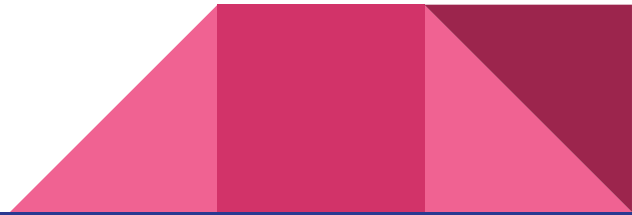
Park

- Open source platform for RL specifically for systems research
- Variety of 12 real-world systems optimisation problems
  - Networking, databases, distributed systems…
  - 7 environments powered by real systems and 5 by simulations
- Framework has one common interface
  - Encourages extension to other systems and problems
- Aimed at machine learning community
  - Don’t have to deal with low-level system implementation; can focus on algorithms

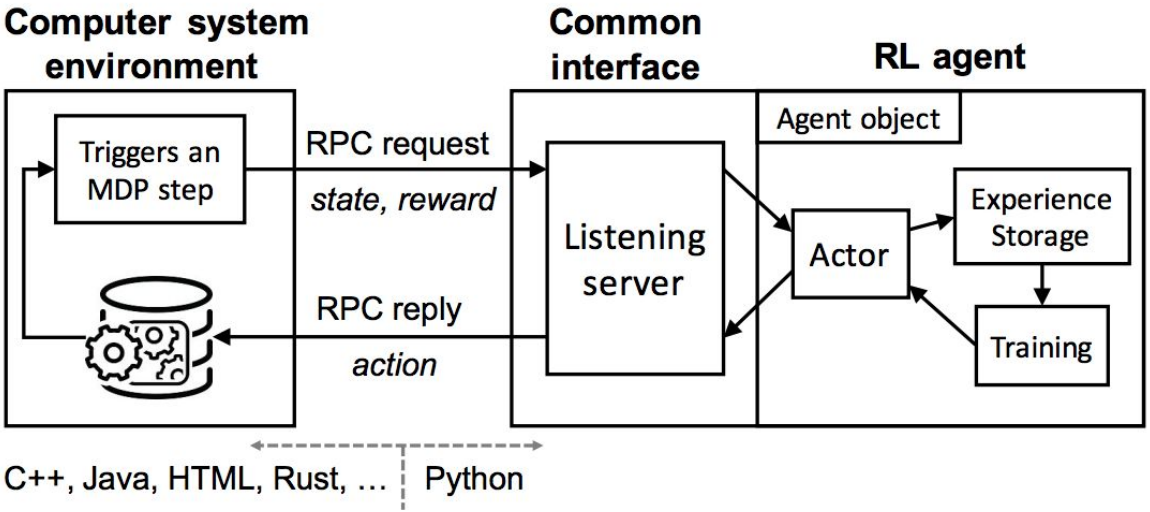


Park

- Markov Decision Process (MDP) formulated for each environment
  - Events triggering MDP step; state and action spaces; reward function
- Backend runs continuously, periodically sends requests to the learning agent to take control actions
- Common interface
  - Easy comparison of different learning agents on a common benchmark
- Server listens for requests from the backend
- Backend and agent are run on different processes (can be different machines)
- Communicate using remote procedure calls



Park



Algorithm 1 Interface for real system interaction.

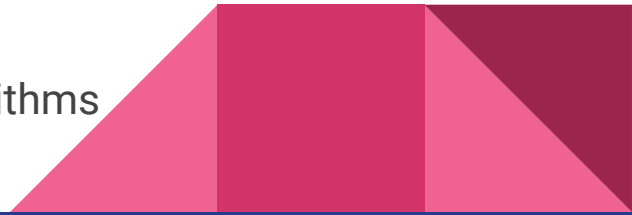
```
def env.run(agent):
    while not done:
        state, reward, done = server.listen()
        # reward for the previous action
        action = agent.act(state, reward, done)
        server.reply(action)
```

Algorithm 2 Interface for simulated interaction.

```
def env.step(action):
    # OpenAI Gym style of interaction
    server.reply(action)
    state, reward, done = server.listen()
    return state, reward, done
```

Experiments & Results

- Existing RL agents trained on 12 Park environments
- Provide heuristics where possible and compare with optimal policy
- Many tasks had to be greatly simplified to apply RL algorithms
  - E.g. Content Delivery Network memory caching uses 1MB cache rather than the typical few GB - causes reward for an action to be significantly delayed
  - E.g. Account region assignment - account only allocated at initialisation, no active migration
- Results for RL algorithms not very good (but this is kind of the point)
  - Focus of paper was to introduce Park, not show off results
  - Existing RL not designed for systems optimisation
  - Shows which systems problems require new RL algorithms
  - All do show improvement over time (sanity check)



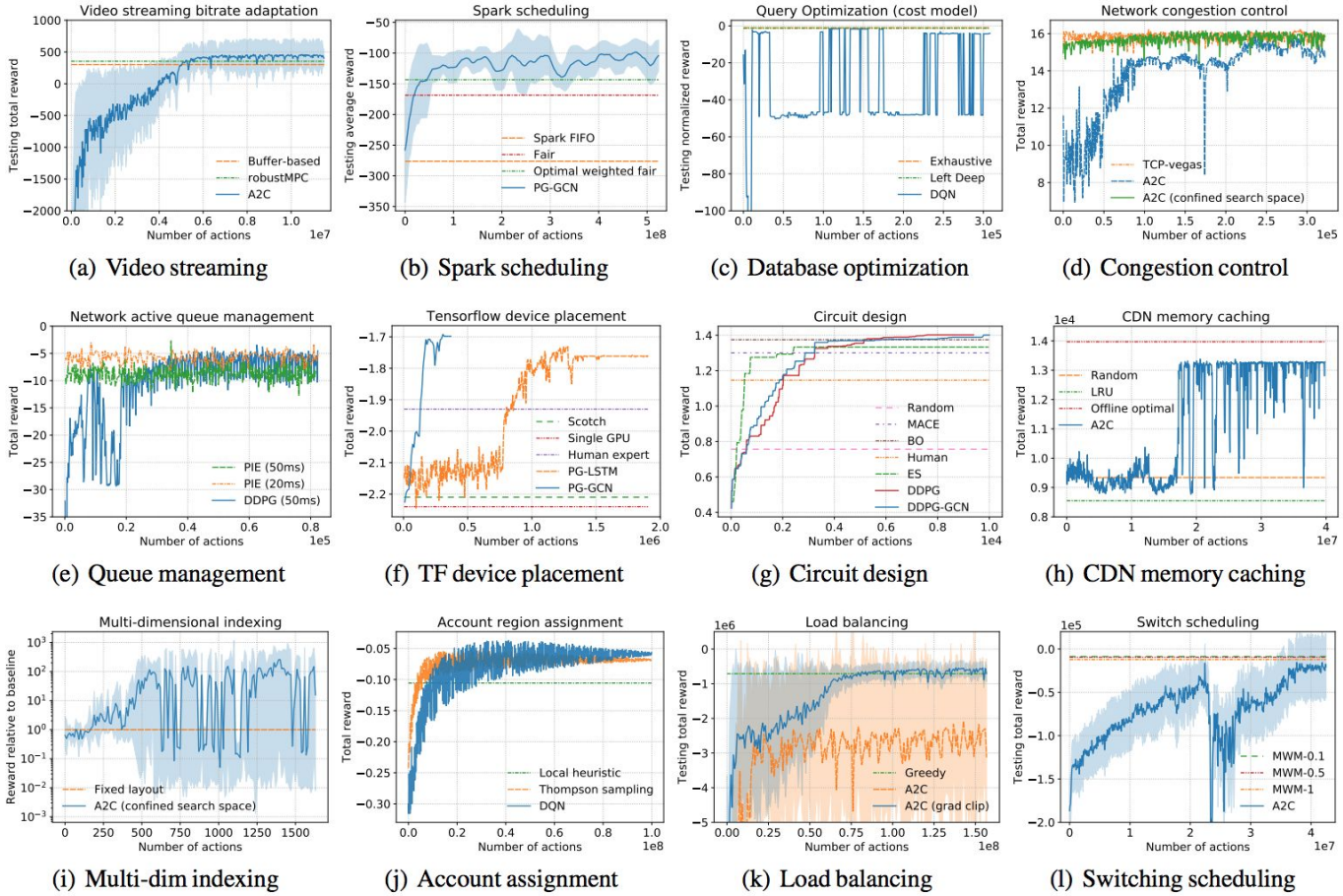


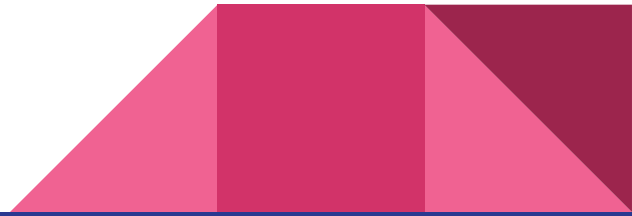
Figure 4. Benchmarks of the existing standard RL algorithms on Park environments. In y-axes, "testing" means the agents are tested with unseen settings in the environment (e.g., newly sampled workload unseen during training, unseen job patterns to schedule, etc.). The heuristic or optimal policies are provided as comparison.

Strengths

- Open Source, freely available
- Scalable
  - Common interface allows multiple concurrent instances, including parallel
- Extensible
  - Simple to define new environments
  - Common interface separates development of RL agent from system implementations
- Easy to use for RL researchers
  - Easy comparison of different learning agents on a common benchmark
- Results illustrate where there is potential further research
  - System problems that may require new RL techniques

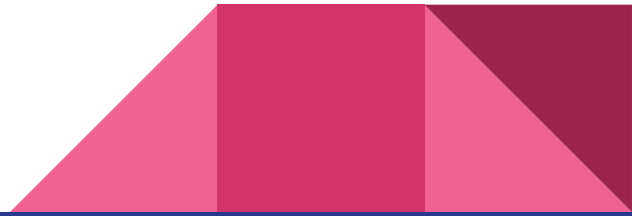
Weaknesses

- Many environments are simulated rather than real
  - Is this sufficient for the experiments? Doesn’t show anything
- Many tasks had to be greatly simplified to apply RL algorithms
  - Is it feasible to use RL for these original tasks?
- Results not discussed - only point out general room for improvement, no specifics
- No mention of how long it takes



Opinion & Impact

- Easy to understand and use system
- Framework and areas of future work provided
  - Opportunity for inventing new learning techniques or altering existing
  - Framework extensible
  - Experiments show areas of improvement - where new algorithms necessary
- Novel idea - could have a large impact



Thanks for listening

Questions?