Learning and Control in Countable State Spaces

R. Srikant
UIUC

Joint work with

Yashaswini Murthy
UIUC

Izzy Grosof
UIUC→Northwestern

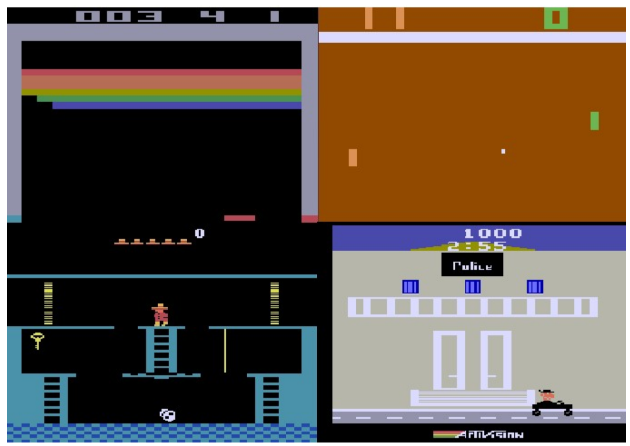
Siva Theja Maguluri
GaTech

- Izzy’s and Yashaswini’s Talks: Room A001, Today 13:30-15:00

Motivation

- Reinforcement Learning (RL)

Go [Silver et al., 2018]



Game playing [Mnih et al., 2013]

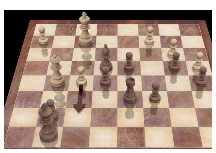
Robotics

Well-Studied and Not So Well-Studied Learning Problems

Finite-State Spaces
$P(s'|s,a)$
(No or very little structure)

Countable State-Spaces
(Limited Structure: positivity, easy stabilizability; how can we exploit it?)

Uncountable, but structured problems
(e.g., LQ/Robust)

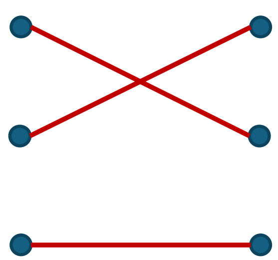


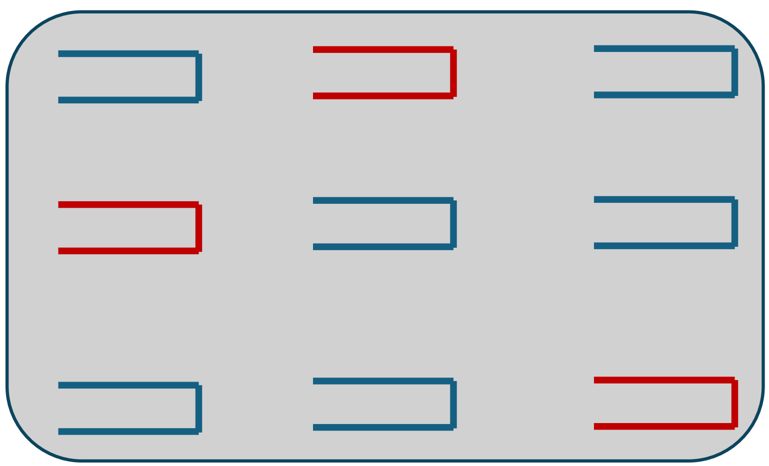
Stability=bounded w.p. 1

$x_{k+1} = Ax_k + Bu_k + w_k$
Either Sys Id followed by Control,
Or Linear Policy Optimization

Scheduling in an nxn Switch

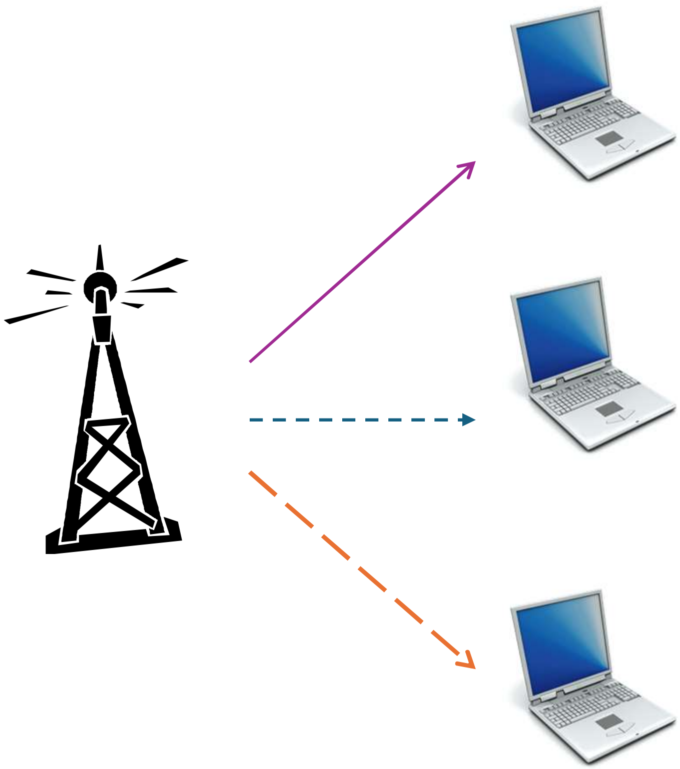
- A matrix of queues operating in discrete-time; packets arrive to each queue according to some arrival process. In each time slot, at most one packet can be served from each queue (Application: Data Center Switches)
- Controls: Permutation matrices
  - At most one queue from each row, and one from each column can be served in each time slot
  - Find a sequence of such matrices to minimize average delay





$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Scheduling in 5G Networks

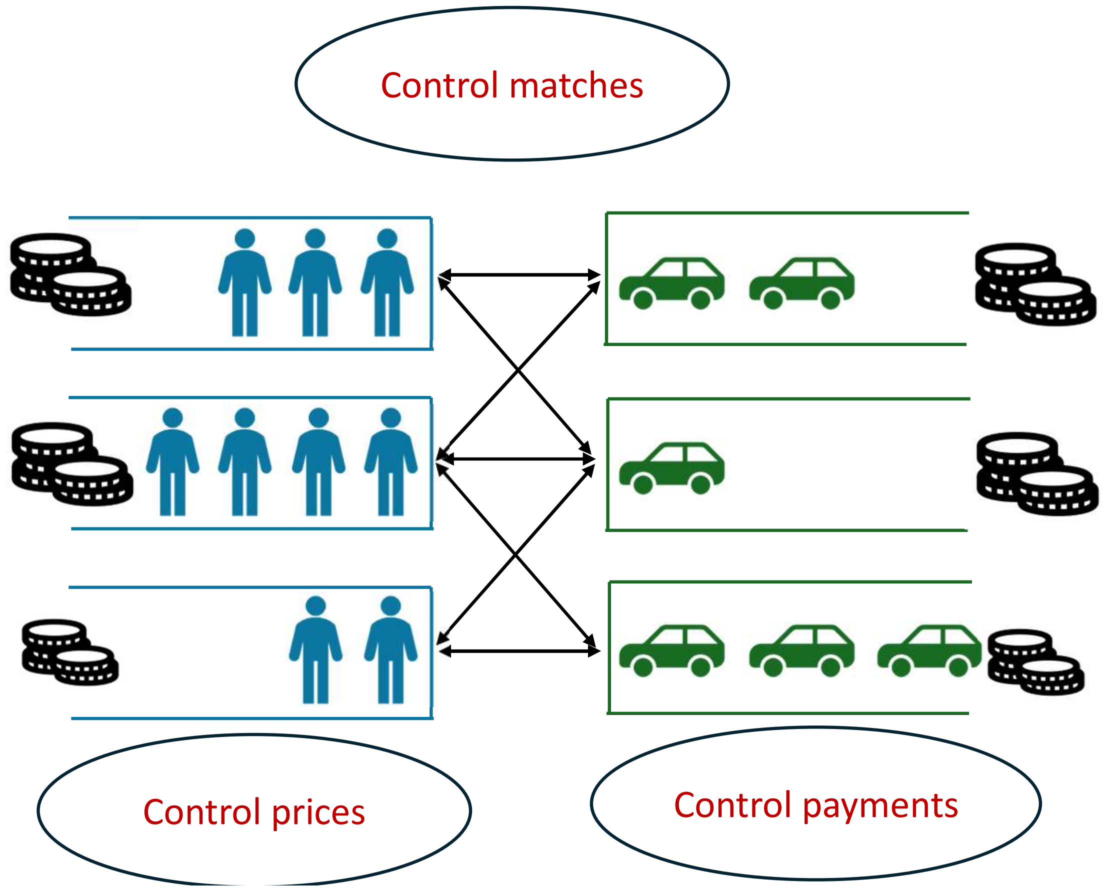


	Time Slot 1	Time Slot 2	.	.	Time Slot T
Freq Slot 1	1	3	3	2	1
Freq Slot 2	1	2	1	2	1
.					
.					
Freq Slot F	3	3	1	2	2

- Available data rate in different slots could be different
- Virtualization requirements: min guarantees on the # slots allocated to different classes of users (police, fire dept, tesla,…)
- Subject to these constraints, minimize delay, ensure fairness, etc.

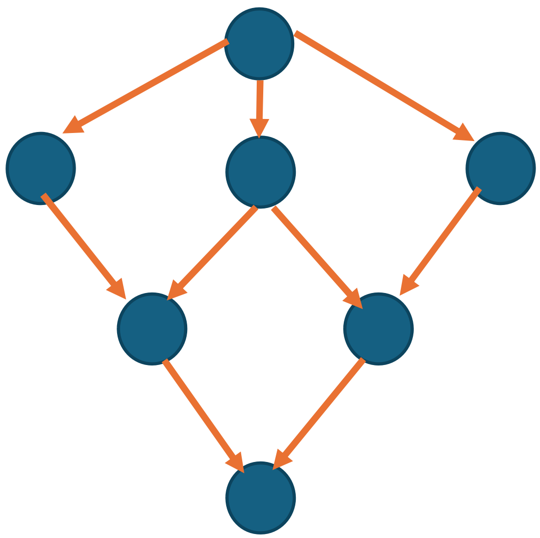
Ride Hailing

- Goal: Maximize revenue minus weighted delay



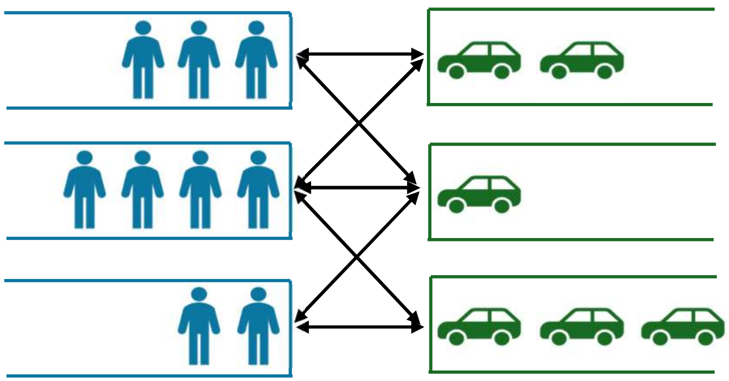
Cloud Computing

- ML jobs may have complicated structures: e.g., DAGs representing precedence among tasks; a sequence of such jobs arrive according to some random process
- Control: allocate tasks to minimize mean job delay



Model

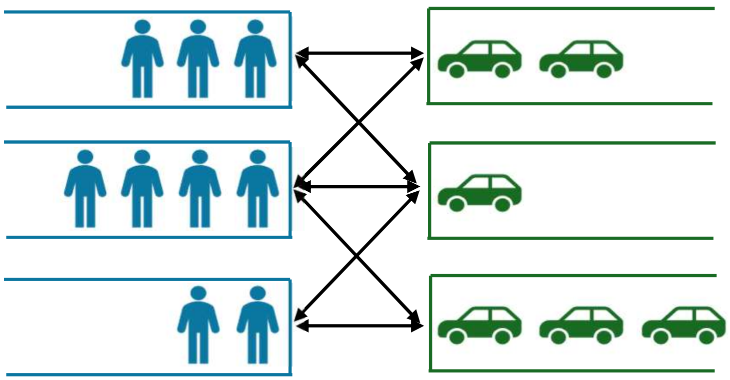
- $s_{k+1} = f(s_k, a_k, w_k)$
  - $s_k \in Z_+^d$ (vector of queue lengths)
  - w_k: randomness
  - $a_k = \pi(s_k)$: (randomized) feedback policy



- $J_\pi = \lim_{T \to \infty} \frac{\mathbb{E}_\pi\left[\sum_{t=0}^{T-1} c(s_k)\right]}{T}$
  - Goal: minimize average cost
  - $c(s_k) = ||s_k||_1$(total queue length)

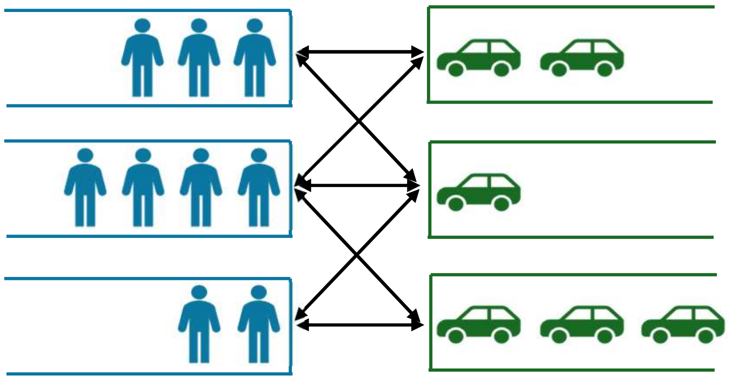
Relative Value Function V

- (Differential) Cost accumulated until you hit 0
  - J_π: Average cost under policy π
  - $||s_k||_1 - J_\pi$: Differential cost
  - τ_s^π time to hit state 0 starting from state s
  - $V_\pi(s) = E(\sum_{k=0}^{\tau_s^\pi}(||s_k||_1 - J_\pi)|s_0 = s)$



Relative Value Function Q

- (Differential) Cost accumulated until you hit 0 if you apply action a in state s and use policy π after that



- $Q_\pi(s,a) \coloneqq$
  $||s||_1 - J_\pi + \mathbb{E}[V_\pi(s_1)|\ s_0 = s, \mathrm{a}_0 = \mathrm{a}]$

Policy Optimization (Policy Iteration)

Policy Improvement

$$\pi(a|s) = \arg\min_{a'} Q(s, a')$$

$$J_\pi + Q(s,a) = c(s) + \mathbb{E}\left[Q\left(s', \pi(s')\right) | s, a\right]$$

(Poisson's Equation)

Policy Evaluation

Policy Optimization (Learning): Natural Policy Gradient

Policy Improvement

NPG update:

$\pi_{k+1}(a|s)$

$\propto \pi_k(a|s)\exp(-\eta \hat{Q}_{\pi_k}(s,a))$

Related to PPO, Dai-Gluzman 2022

Collect samples from a trajectory generated by π_k and estimate $\hat{Q}_{\pi_k}$

Policy Evaluation

Abstracting Policy Evaluation (Ignoring TD Learning, etc.)

Policy Improvement

NPG update:

$$\pi_{k+1}(a|s)$$

$$\propto \pi_k(a|s)\exp(-\eta \hat{Q}_{\pi_k}(s,a))$$

$$||Q_{\pi_k} - \hat{Q}_{\pi_k}||_\infty \leq \kappa$$

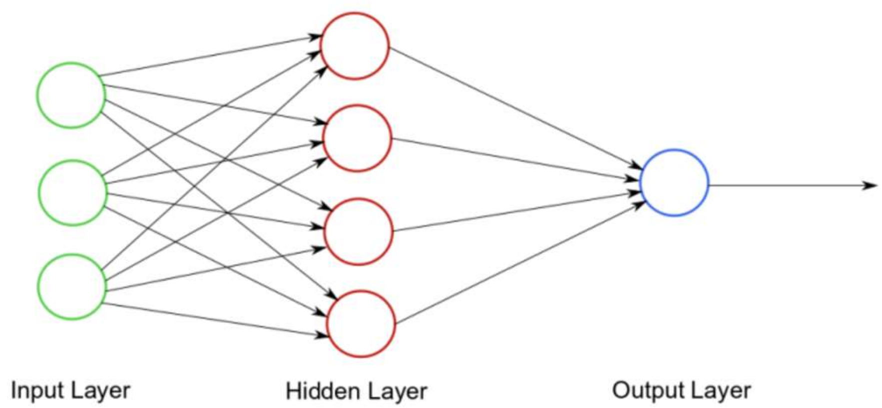
(function approximation error)

not a good abstraction, a better abstraction later

Policy Evaluation

Rationale for the Abstraction

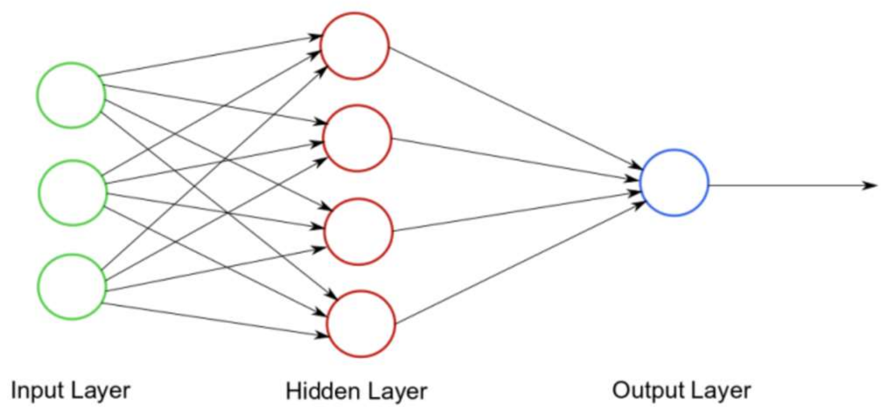
- What can neural networks do?



- Input: (s, a), Output: $\hat{Q}(s, a)$
- The reason we need a neural network is that we cannot visit all (state,action) pairs: so, visit a few (state, action) pairs, empirically estimate the Q-function at those (state,action) pairs and extrapolate to other (state,action) pairs by training a neural network

Rationale for the Abstraction

- What can neural networks do?



- Input: (s, a), Output: $\hat{Q}(s, a)$

- Our abstraction assumes that the neural network can uniformly approximate the Q-value at all (state,action) pairs: not reasonable for countable state spaces, but we will stick with this assumption for now and refine it later

Result for Finite-State Spaces (Even-Dar et al, 2009, Abbasi-Yadkori et al, 2019)

- Consider T iterations of the Natural Policy Gradient algorithm.
- Step size $(\hat{Q}_{\max} = \max_{s,a,k} \hat{Q}_{\pi_k}(s,a))$

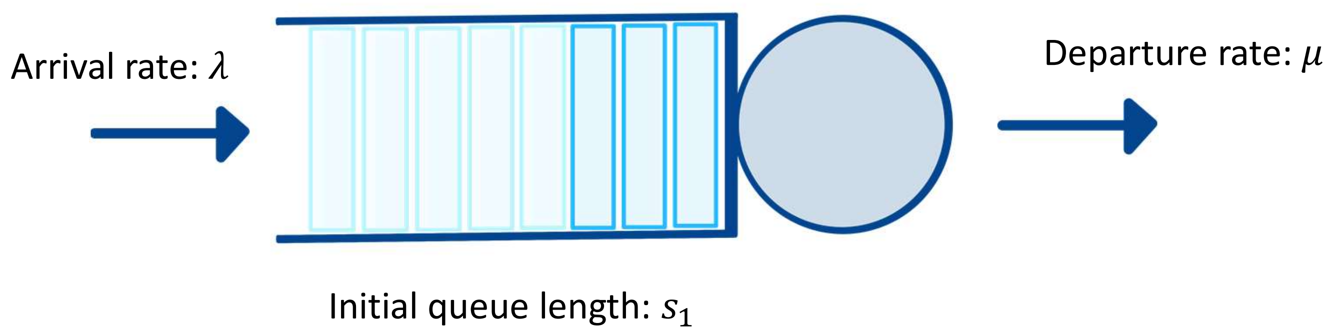
$$\eta = \sqrt{\frac{8 \log |A|}{T \hat{Q}_{max}}}$$

- The overall regret (up to a function approximation error):

$$\sum_{k=1}^{T} J_{\pi_k} - J_{\pi^*} \leq O\left(\sqrt{T} \hat{Q}_{max}\right)$$

What is $Q_\pi(s, a)$?

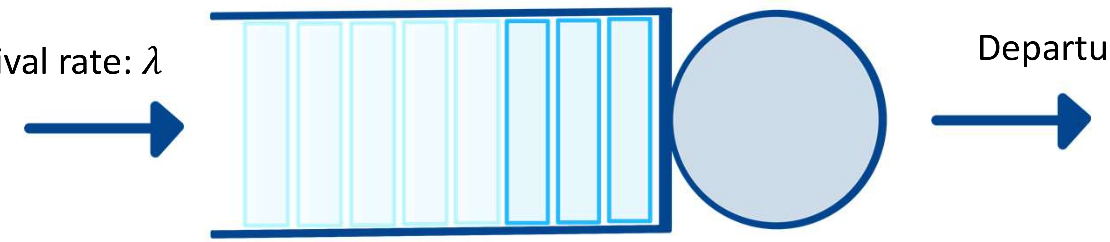
- Recall $Q_\pi(s,a) \coloneqq ||s||_1 - J_\pi + \mathbb{E}[V_\pi(s_1)|\ s_0 = s, \mathrm{a}_0 = \mathrm{a}]$
- $V_\pi(s_1)$: (relative) cost accumulated until you hit 0 starting from s_1
- Therefore, $V_\pi(s_1) \propto ||s_1||^2$. Why?
- Fluid Intuition:



- Cost at time t is $s_1 - (\mu - \lambda)t \ \Rightarrow\ V(s_1) \propto s_1^2$

What is $Q_\pi(s, a)$?

- Fluid Intuition:



- Cost at time t is $s_1 - (\mu - \lambda)t \quad \Rightarrow \quad V(s_1) \propto s_1^2$
- For this intuition, the system should be stable: in this simple single-queue case $\lambda < \mu$.
  - This observation about stability will be useful later

Countable State Spaces

- The overall regret (up to a function approximation error):

$$\sum_{k=1}^{T} J_{\pi_k} - J_{\pi^*} \leq O\left(\sqrt{T}\hat{Q}_{max}\right)$$

- Even if the system is stable at each iteration of NPG (which it is not obvious that it will be), perhaps one can show that $Q_{\pi_k}(s,a) \propto ||s||^2$, but that doesn't help: $\hat{Q}_{max} = \infty$

What Goes Wrong and How to Fix It?

NPG update:

$\pi_{k+1}(a|s)$

$\propto \pi_k(a|s)\exp(-\eta \hat{Q}_{\pi_k}(s,a))$

$||Q_{\pi_k} - \hat{Q}_{\pi_k}||_\infty \leq \kappa$

(function approximation error)

- η is like a step-size (algorithm is closely related to mirror descent)
  - It has to be sufficiently small in magnitude, of the order of $1/\hat{Q}_{max}$ but the error is proportional to $1/\eta$
- Solution: Make η state-dependent but proportional to $1/Q_{max}(s)$
- To be able to this, we need an estimate of $\mathrm{Q}_{\max}(\mathrm{s}) \coloneqq \max_a Q(s,a)$

Roadmap to Handle Countable State Spaces

- Use the structure of our motivating examples to ensure that the system is always stable under any policy generated by NPG (i.e., bounded w.p. 1)
  - Tradeoff between robustness and performance

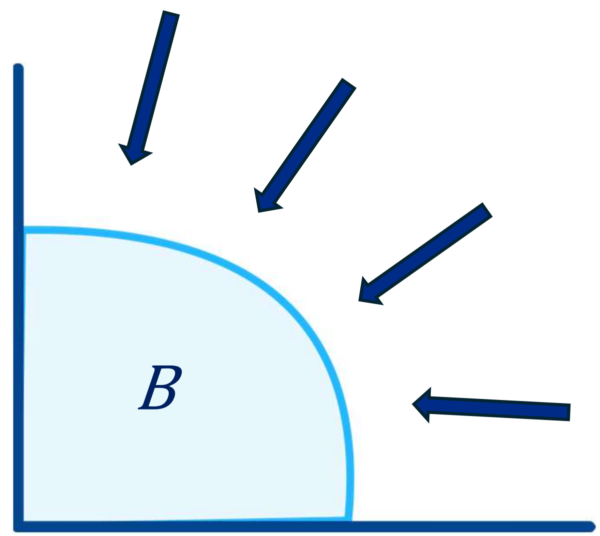
- This will allow us to show that $Q(s) \leq c_1||s||^2 + c_2||s||_1 + c_3$

- Make a small change to the algorithm to exploit the above bound on $Q(s)$ and eliminate the dependence on Q_{max}

Exploiting Structure: Drift Assumption

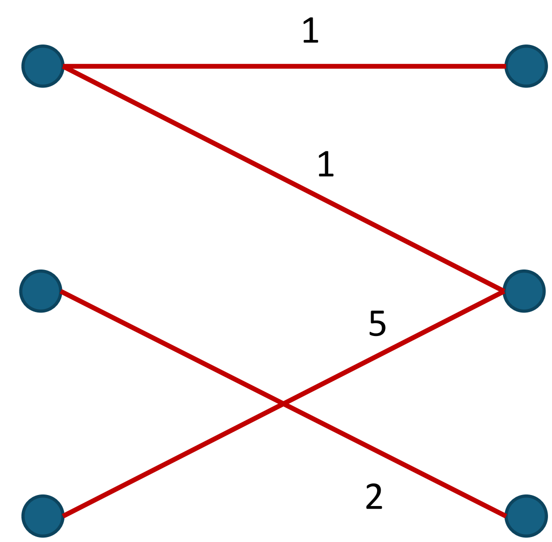
$$\mathbb{E}_\pi[||s_{k+1}||^2 - ||s_k||^2 | s_k = s] \leq -\epsilon ||s||_1 + c \qquad \forall \pi, s$$

- We will only search among the class of controls that satisfy the above Lyapunov drift condition
  - See also (Xie, Shah, Xu, 2020), (Lale et al, 2023)
- Question: can we assure such robust stability?
  - Naturally satisfied in some cases, e.g., with abandonments
  - In other cases, we may give up some performance (although probably doesn't matter in practice) for robust stability, i.e., independent of problem parameters



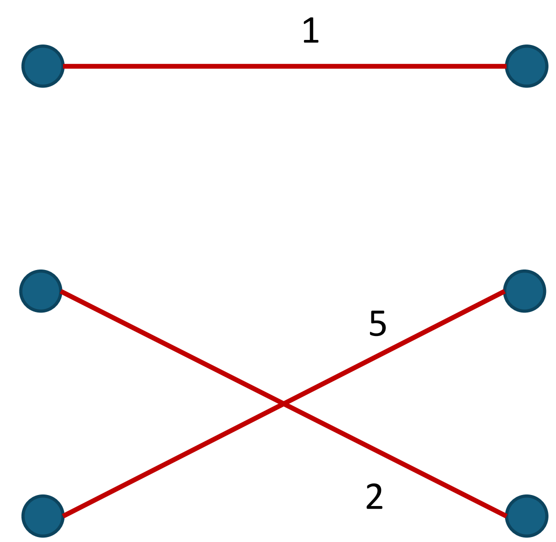
MaxWeight Algorithm (Example)

- Bipartite graph: weight of an edge from node i to node j on the right equal to q_{ij}



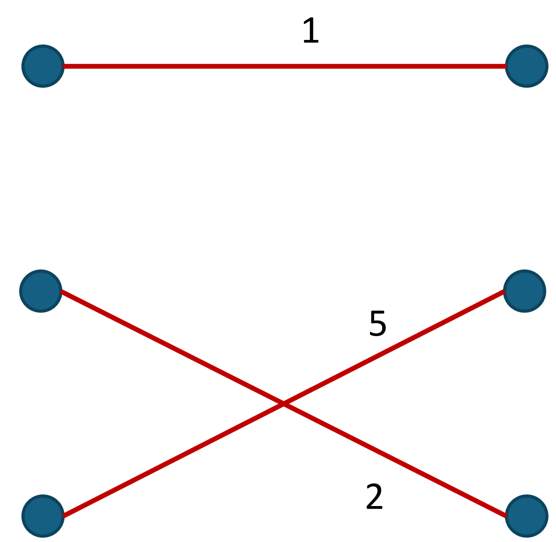
MaxWeight Algorithm (Example)

- Bipartite graph: weight of an edge from node i to node j on the right equal to q_{ij}
- Find a matching with the largest weight
- This algorithm always stabilizes the system if the system is stabilizable
- But this may not be optimal



MaxWeight Algorithm (Example)

- Bipartite graph: weight of an edge from node i to node j on the right equal to q_{ij}
- Find a matching with the largest weight
- Solution: Use this algorithm with low probability when the queue lengths are small and use with higher and higher probability when the queue length gets larger



Satisfying the Drift Assumption: Soft Thresholding

$$\pi(s) = \begin{cases} \pi_{NPG}(s) & \text{w.p. } \min\left(1, \frac{1}{\lambda||s||}\right) \\ \pi_{MW}(s) & \text{w.p. } 1 - \min\left(1, \frac{1}{\lambda||s||}\right) \end{cases}$$

- When $||s|| \geq \frac{1}{\lambda}$ the soft thresholding begins.
- At larger queue lengths, MaxWeight policy dominates, ensuring stability
- At lower queue lengths, NPG dominates which focuses on optimality
- Hence thresholding provides an optimality-stability tradeoff

Value Function under the Drift Assumption

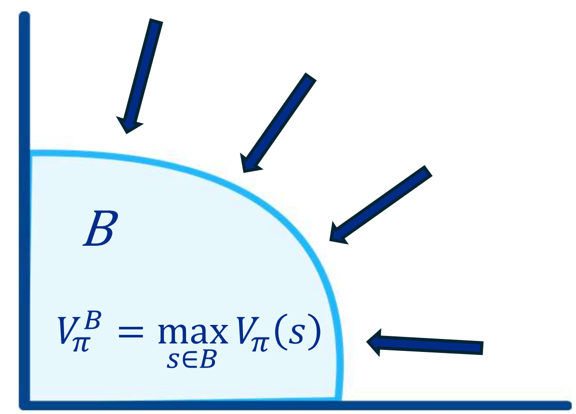
- Recall the drift equation:

$$\mathbb{E}_\pi[||s_{k+1}||^2 - ||s_k||^2 | s_k = s] \leq -\epsilon ||s||_1 + c \qquad \forall \pi, s$$

- Using this inequality, one can show

$$V_\pi(s) \leq \frac{2}{\epsilon}||s||^2 + V_\pi^B \qquad \forall s \neq B, \forall \pi$$

Not policy independent!



- Recall the fluid intuition from before for the first term.

- But what about the second term?

Exploit Additional Structure

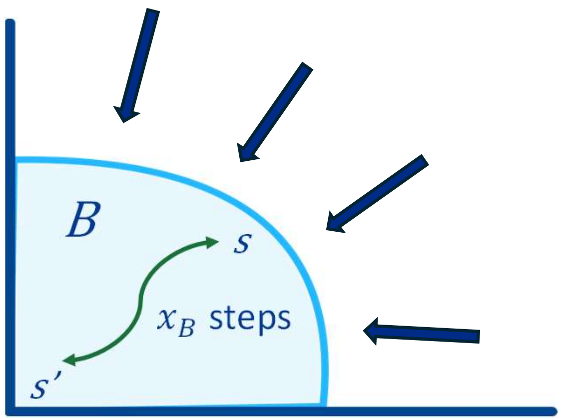
- Bounded arrivals and bounded departures i.e.,

$$\mathbb{P}_\pi(s'|s) > 0 \Rightarrow ||s'||^2 \leq c_1||s||^2 + c_2 \;\; \forall \pi$$

- Can move from any state $s \in B$ to any state $s' \in B$

  in at most x_B time slots with probability at least p_B, i.e.,

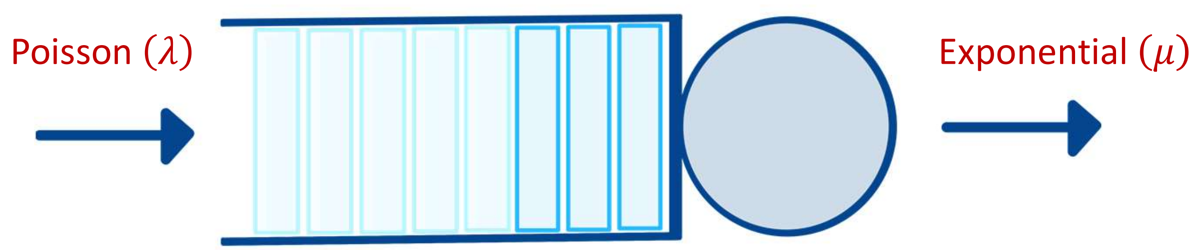
$$\mathbb{P}_\pi^{x_B}(s'|s) \geq p_B \quad \forall s, s' \in B, \pi$$



Exploiting Problem Structure

- Assumption: within a finite set there is non-zero probability of moving from any $s \in B$ to $s' \in B$ in a finite amount of time with non-zero probability

- For instance, consider a simple M/M/1 queue



- For any $s \leq B$, there is a non-zero probability to hit state 0 from s i.e., when no arrivals occur. It is also possible to move from 0 to any state $s \leq B$ with non-zero probability ie., when no departures occur.

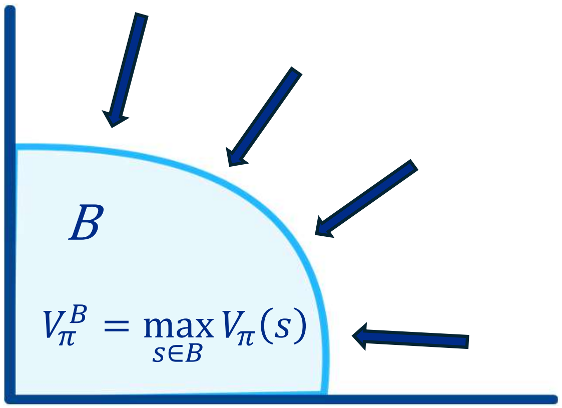
Implications of the Structural Assumption

- Uniform upper bound on the value function for all

policies $\pi \in \Pi$, for all states within B i.e.,

$$\max_{\pi \in \Pi} V_\pi^B \leq \frac{c}{p_B \epsilon}\left(\frac{x_B}{p_B^2} + 2x_B\right)$$

- The bounds on V_π are obtained by studying the solution to Poisson's equation and obtaining robust bounds
  - See Glynn-Meyn (1996) for policy dependent bounds



A Key Result

If all policies $\pi \in \Pi$ induce a Markov chain that:

- is irreducible
- satisfies the drift equation
- satisfies the additional structural assumption

then, the state action value function Q_π can be uniformly bounded:

$$Q_\pi(s,a) \leq \frac{2}{\epsilon} c_1 ||s||^2 + C$$

State Dependent,
policy-independent
upper bound

Why does it matter?

Policy Improvement

NPG update:

$\pi(a|s)$

$\propto \pi(a|s)\exp(-\boldsymbol{\eta_s}\hat{Q}_\pi(s,a))$

Allows us to choose the step-size as a function of s

$||Q_\pi - \hat{Q}_\pi||_\infty \leq \kappa$

(function approximation error)

Policy Evaluation

Theorem (Learning and Control in Queues)

Set $\eta_s = \sqrt{\frac{8 \log |A|}{T} \frac{1}{M_s}}$, where M_s is a quadratic in s.

Up to function approximation error:

$$\sum_{k=1}^{T} (J_{\pi_k} - J_{\pi^*}) \leq c' \sqrt{T}$$

With the function approximation error:

$$\sum_{k=1}^{T} (J_{\pi_k} - J_{\pi^*}) \leq \kappa T + c' \sqrt{T}$$

$\kappa \rightarrow 0$ as the number of neurons in the neural network goes to infinity

Recall our Abstraction of Policy Evaluation

Policy Improvement

NPG update:

$$\pi(a|s) \propto \pi(a|s)\exp(-\eta_s \hat{Q}_\pi(s,a))$$

$$||Q_\pi - \hat{Q}_\pi||_\infty \leq \kappa$$

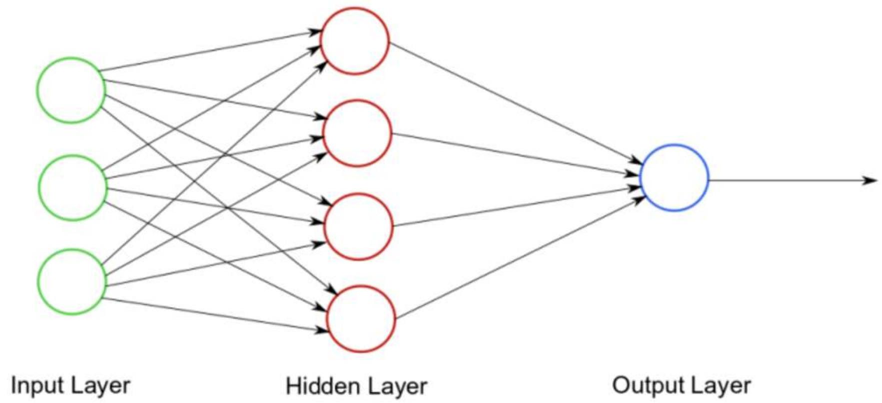
(function approximation error)

Policy Evaluation

This abstraction is problematic

Comments on the Policy Evaluation abstraction

- What can neural networks do?



- Universal approximation theorem: Sufficiently large neural networks can approximate any continuous function on a compact domain to an arbitrary degree of accuracy (an explicit construction in (Satpathi-S., 2019))

Comments on the Policy Evaluation abstraction

- However, our function $Q(s) \sim ||s||^2$
- The domain (being countable) can be compactified, but the function blows up to infinity, so neural networks cannot uniformly approximate $Q(s)$
- On the other hand

$$\frac{Q(s)}{||s||^2}$$

is bounded, so a modified abstraction is more reasonable (but there is more to be proved here)

A Modified Abstraction of Policy Evaluation

Policy Improvement

NPG update:

$$\pi(a|s) \propto \pi(a|s)\exp(-\eta_s \hat{Q}_\pi(s,a))$$

$$|Q_\pi(s,a) - \hat{Q}_\pi(s,a)| \le \kappa||s||^2$$

(function approximation error)

Policy Evaluation

Theorem (Learning and Control in Queues)

Set $\eta_s = \sqrt{\frac{8 \log |A|}{T} \frac{1}{M_s}}$, where M_s is a quadratic in s.

Up to function approximation error:

$$\sum_{k=1}^{T} (J_{\pi_k} - J_{\pi^*}) \leq c' \sqrt{T}$$

With the function approximation error:

$$\sum_{k=1}^{T} (J_{\pi_k} - J_{\pi^*}) \leq \kappa \max_{\pi} E_{\pi}(||s||^2)\, T + \; c' \sqrt{T}$$

$\kappa \rightarrow 0$ as the number of neurons goes to infinity;
But is this a meaningful result?

Is $\max_{\pi} E_{\pi}(||s||^2)$ finite?

- Recall the drift equation:

$$\mathbb{E}_{\pi}[||s_{k+1}||^2 - ||s_k||^2 | s_k = s] \leq -\epsilon ||s||_1 + c \qquad \forall \pi, s$$

- One can show that this ensures that the moments of $||s||$ exist (Eryilmaz, S., 2012), (Hajek 1982), i.e., there exists α such that,

$$\mathbb{E}_{\pi}\left[e^{\alpha ||s||}\right] \leq M,$$

  independent of π

- This implies $\max_{\pi} E_{\pi}(||s||^2)$ is finite

Thus, we have a valid Theorem

Set $\eta_s = \sqrt{\frac{8 \log |A|}{T} \frac{1}{M_s}}$, where M_s is a quadratic in s.

Up to function approximation error:

$$\sum_{k=1}^{T} (J_{\pi_k} - J_{\pi^*}) \leq c' \sqrt{T}$$

With the function approximation error:

$$\sum_{k=1}^{T} (J_{\pi_k} - J_{\pi^*}) \leq \kappa \max_{\pi} E_{\pi}(||s||^2)\, T + c' \sqrt{T}$$

$\kappa \to 0$ as the number of neurons goes to infinity

Conclusions

- Exploited problem structure to design RL algorithms to control countable state-space applications like communication networks, cloud computing, and ride sharing
  - Even when the problem has very limited structure

- Key Algorithmic Idea: Use state-dependent step-sizes in the policy improvement part of the NPG algorithm

- Key Proof Idea: Bound the relative value function (also called the solution to Poisson's equation) and relate it learning-theoretic ideas in prediction-from-expert advice a.k.a. online mirror descent (in prior work on finite-state spaces)

Thank You!