Optimal Trunk Reservation by Policy Learning[1]

Antonio Massaro [1] Francesco De Pellegrini [1, 2] Lorenzo Maggi [3]

[1] Nokia Bell Labs, Paris. Fondazione Bruno Kessler, Trento, Italy

[2]University of Avignon, France

[3]Nokia Bell Labs, Paris

12/04/2019

[1]INFOCOM 2019, Paris

Integer Gradient Ascent

A **reinforcement learning** algorithm for optimal **admission control** for a **queue** with **finite buffer** and different **jobs' priority levels**.

The model

- D job types, M memory slots
- Independent Poisson arrivals: $\lambda_1, ..., \lambda_D$
- Poisson service time μ
- States: $(m, d) \in \{0, ..., M\} \times \{1, ..., D\}$
- Actions: $A(m, d) = \{0, 1\}$, $A(M, d) = \{0\}$
- Rewards: $r_1 > r_2 > ... > r_D$.
- Policy: $\pi_t(m, d) = \mathbb{P}(a(m, d) = 1)$

$$\max_{\pi} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}[r_{d_t} \pi_t(m_t, d_t)].$$

Solve it on-line, without knowledge on flow arrivals distribution and service time

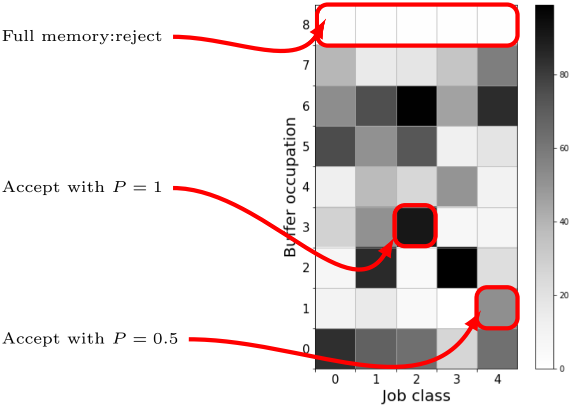
How does an optimal policy look like?

Finite states and actions + bounded rewards + unichain:

$\exists$ optimal stationary policy[1].

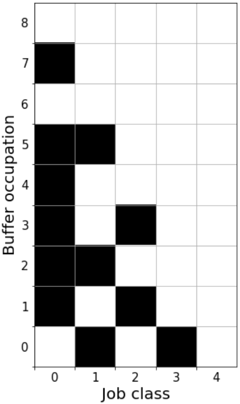
How does an optimal policy look like?

Finite states and actions + bounded rewards + unichain:
$\exists$ optimal stationary policy[1].



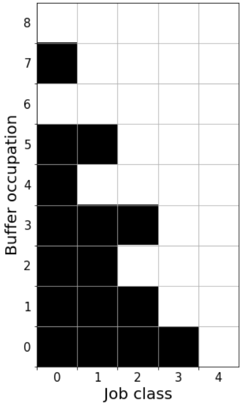
How does an optimal policy look like?

- Stationary
- Deterministic



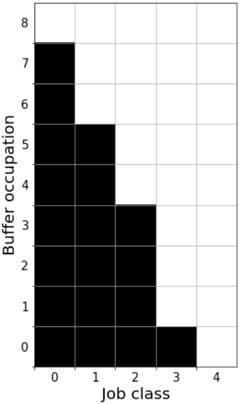
How does an optimal policy look like?

- Stationary
- Deterministic
- Threshold



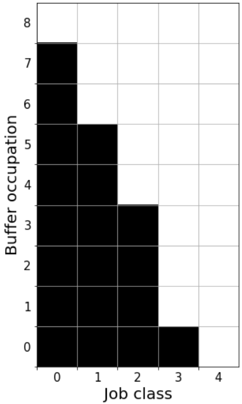
An optimal policy looks like this

- Stationary
- Deterministic
- Threshold
- Monotonic[2]



An optimal policy looks like this

- Stairway policy



Optimal policy calculation, partial knowledge

An optimal policy can be calculated by

- Q-learning
- Policy iteration

Can we do better?

Specialize the search algorithm to the structure of the optimal policy.

Optimal policy calculation, partial knowledge

An optimal policy can be calculated by

- Q-learning
- Policy iteration

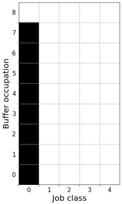
Can we do better?

Specialize the search algorithm to the structure of the optimal policy.

Policy search: basic idea

Idea: progressively 'fill' the probability of admission at each state

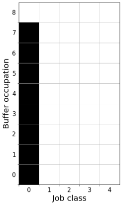
0



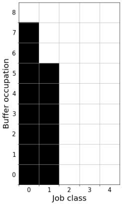
Policy search: basic idea

Idea: progressively 'fill' the probability of admission at each state

0



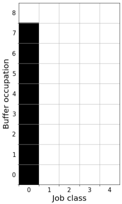
1



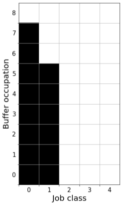
Policy search: basic idea

Idea: progressively 'fill' the probability of admission at each state

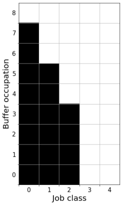
0



1



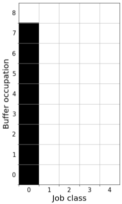
2



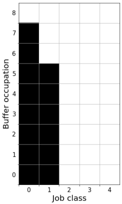
Policy search: basic idea

Idea: progressively 'fill' the probability of admission at each state

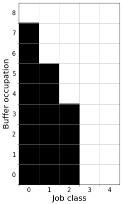
0



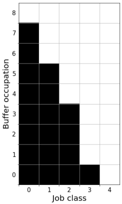
1



2



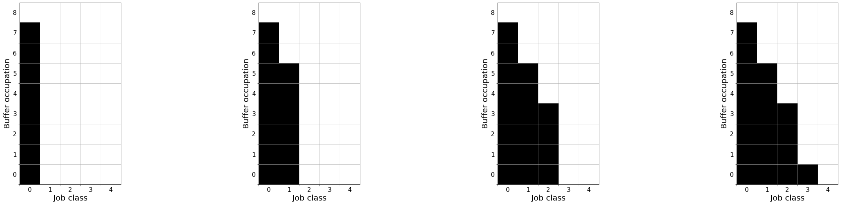
3



Policy search: basic idea

Idea: progressively ’fill’ the probability of admission at each state

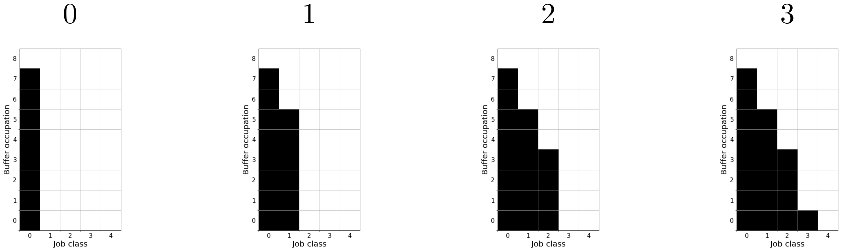
0 1 2 3



How do we do it?

Policy search: basic idea

Idea: progressively 'fill' the probability of admission at each state

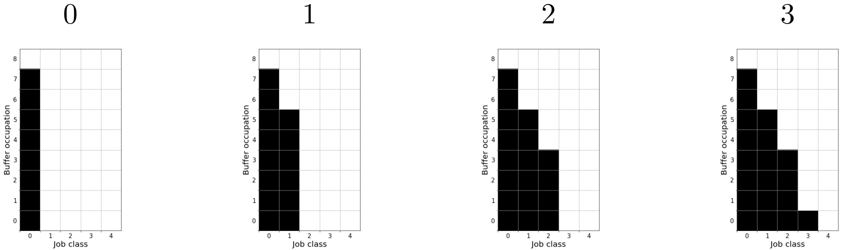


How do we do it?

- We are maximizing the average reward

Policy search: basic idea

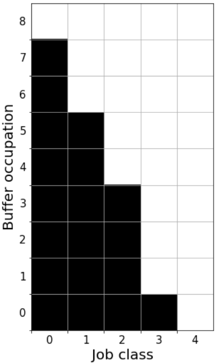
Idea: progressively 'fill' the probability of admission at each state



How do we do it?

- We are maximizing the average reward
- We need "local" information on the average reward: a gradient

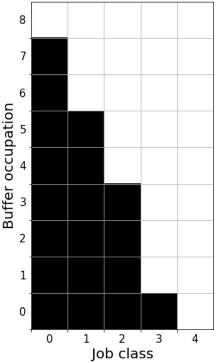
A differentiable parametrization for policies



$$\leftrightarrow \qquad (4,3,3,3,2,2,1,1) \in [0,5]^7$$

$$(\theta_1, ..., \theta_M) \in [0, D]^M$$

A differentiable parametrization for policies



$\leftrightarrow$ $\quad (4, 3, 3, 3, 2, 2, 1, 1) \in [0, 5]^7$

$(\theta_1, ..., \theta_M) \in [0, D]^M$

A differentiable parametrization of the average reward

$$\rho(\pi) : \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} \mathbb{E}_\pi[r(m_i, d_i)]$$

$$\rho \circ \pi : [0, D]^M \longrightarrow \Gamma \longrightarrow \mathbb{R}$$

$$(\theta_1, ..., \theta_M) \mapsto \pi_\theta \mapsto \rho(\pi_\theta)$$

Take the gradient and do gradient ascent!

A differentiable parametrization of the average reward

$$\rho(\pi) : \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} \mathbb{E}_{\pi}[r(m_i, d_i)]$$

$$\rho \circ \pi : [0, D]^M \longrightarrow \Gamma \longrightarrow \mathbb{R}$$

$$(\theta_1, ..., \theta_M) \mapsto \pi_\theta \mapsto \rho(\pi_\theta)$$

Take the gradient and do gradient ascent!

A formula for the gradient of the average reward[3]

$$\nabla_\theta \rho = \sum_{s \in S} p^\pi(s) \sum_{a \in A(s)} \nabla_\theta \pi(a|s) Q^\pi(s,a)$$

- $s \in S$: state space
- $a \in A(s)$: actions space at state s
- $p^\pi(s)$: stationary probability of state s under policy π
- $Q^\pi(s,a) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = s, a_0 = a\right]$

In our case it is very simple!

$$\frac{\partial \rho}{\partial \theta_m} = p^\pi(m, \lfloor \theta_m \rfloor)(Q^\pi(m, \lfloor \theta_m \rfloor, 1) - Q^\pi(m, \lfloor \theta_m \rfloor, 0)).$$

A formula for the gradient of the average reward[3]

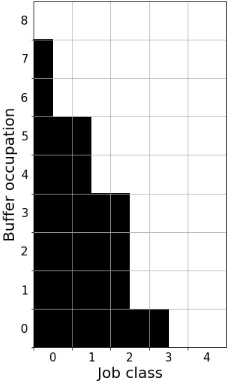
$$\nabla_\theta \rho = \sum_{s \in S} p^\pi(s) \sum_{a \in A(s)} \nabla_\theta \pi(a|s) Q^\pi(s,a)$$

- $s \in S$: state space
- $a \in A(s)$: actions space at state s
- $p^\pi(s)$: stationary probability of state s under policy π
- $Q^\pi(s,a) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = s, a_0 = a\right]$

In our case it is very simple!

$$\frac{\partial \rho}{\partial \theta_m} = p^\pi(m, \lfloor \theta_m \rfloor)(Q^\pi(m, \lfloor \theta_m \rfloor, 1) - Q^\pi(m, \lfloor \theta_m \rfloor, 0)).$$

The gradient of the average reward

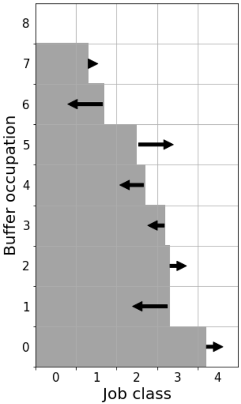


$$\leftrightarrow \quad \pi = (3.5, 2.5, 2.5, 2.5, 1.5, 1.5, .5, .5)$$

$$\frac{\partial \rho}{\partial \theta_0} = p^\pi(0,3)(Q^\pi(0,3,1) - Q^\pi(0,3,0)).$$

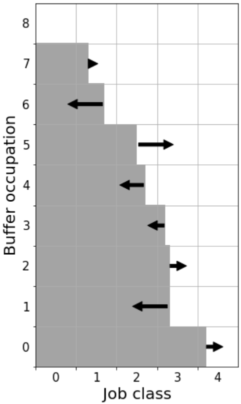
Gradient ascent

1: evaluate the gradient

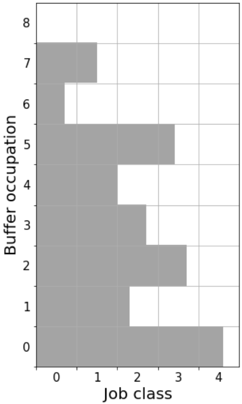


Gradient ascent

1: evaluate the gradient

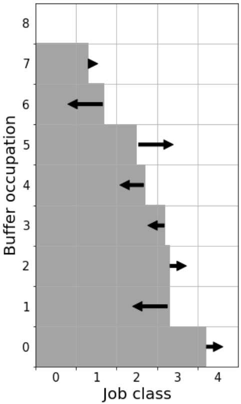


2: move the thresholds

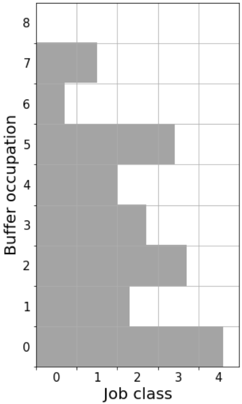


Gradient ascent

1: evaluate the gradient



2: move the thresholds



Warning!

Gradient ascent

Problems:

1. An optimal policy is deterministic
   - Gradient ascent searches among all threshold policies
2. The gradient is discontinuous at deterministic policies
   - $\partial_m^- \rho = p^\pi(m, \theta_m)(Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0))$.
   - $\partial_m^+ \rho = p^\pi(m, \theta_m + 1)(Q^\pi(m, \theta_m + 1, 1) - Q^\pi(m, \theta_m + 1, 0))$.

Solutions:

1. Take integer steps to explore just integer policies.
2. Calculate both gradients and use them in the update step.

Gradient ascent

Problems:

1. An optimal policy is deterministic
   - Gradient ascent searches among all threshold policies
2. The gradient is discontinuous at deterministic policies
   - $\partial_m^- \rho = p^\pi(m, \theta_m)(Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0))$.
   - $\partial_m^+ \rho = p^\pi(m, \theta_m + 1)(Q^\pi(m, \theta_m + 1, 1) - Q^\pi(m, \theta_m + 1, 0))$.

Solutions:

1. Take integer steps to explore just integer policies.
2. Calculate both gradients and use them in the update step.

Gradient ascent

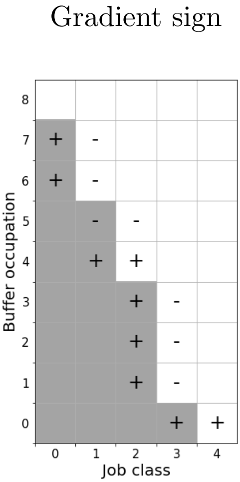
Problems:

1. An optimal policy is deterministic
   - Gradient ascent searches among all threshold policies
2. The gradient is discontinuous at deterministic policies
   - $\partial_m^- \rho = p^\pi(m, \theta_m)(Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0))$.
   - $\partial_m^+ \rho = p^\pi(m, \theta_m + 1)(Q^\pi(m, \theta_m + 1, 1) - Q^\pi(m, \theta_m + 1, 0))$.

Solutions:

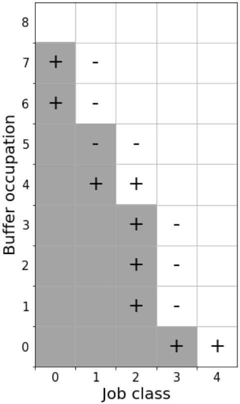
1. Take integer steps to explore just integer policies.
2. Calculate both gradients and use them in the update step.

Integer Gradient Ascent: the idea

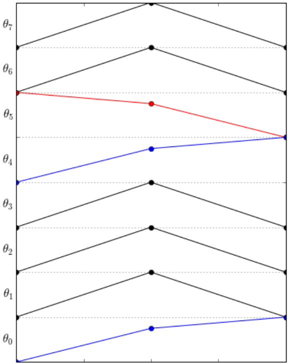


Integer Gradient Ascent: the idea

Gradient sign

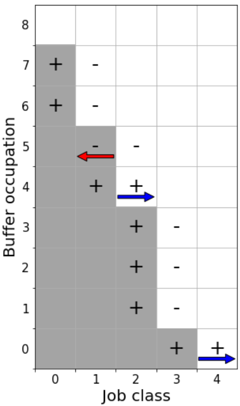


Average reward reaction

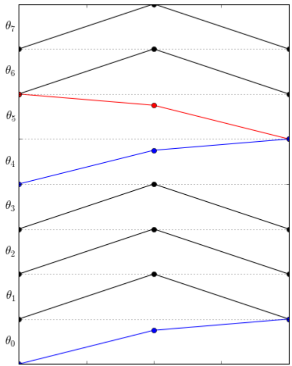


Integer Gradient Ascent: the idea

Gradient sign

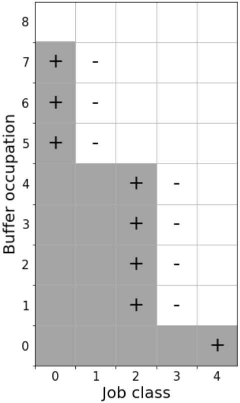


Average reward reaction

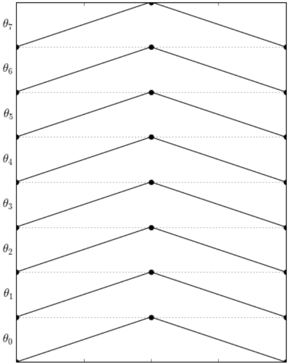


Integer Gradient Ascent: the idea

Gradient sign



Average reward reaction



IGA is correct

Theorem

IGA converges to an optimal policy in a finite number of steps

Proof.

1. At each step at least one threshold is modified
2. At each step the value of at least one state strictly increases
3. MDP is finite, policies are finite $\Rightarrow$ values cannot increase forever
4. IGA must stop
5. At termination, π satisfies Bellman equation, hence it is optimal

$\square$

Ok, but we still have to estimate the gradient!

IGA is correct

Theorem

IGA converges to an optimal policy in a finite number of steps

Proof.

1. At each step at least one threshold is modified
2. At each step the value of at least one state strictly increases
3. MDP is finite, policies are finite $\Rightarrow$ values cannot increase forever
4. IGA must stop
5. At termination, π satisfies Bellman equation, hence it is optimal

$\square$

Ok, but we still have to estimate the gradient!

Gradient estimation

On line gradient estimation

$$\partial_m \rho = p^\pi(m, \theta_m)(Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0))$$

We just need the *sign*

$$Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0) > 0 \ ?$$

How to estimate it?

Gradient estimation

On line gradient estimation

$$\partial_m \rho = p^\pi(m, \theta_m)(Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0))$$

We just need the *sign*

$$Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0) > 0 \ ?$$

How to estimate it?

State-action value estimation

$$Q^\pi(m, \theta_m, 1) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

Estimate it by sampling! Sample an infinite sum?

$$\sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] = \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] + o(T)$$

$$Q^\pi(m, \theta_m, 1) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

But we don't know $\rho(\pi)$!

State-action value estimation

$$Q^\pi(m, \theta_m, 1) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

Estimate it by sampling! Sample an infinite sum?

$$\sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] = \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] + o(T)$$

$$Q^\pi(m, \theta_m, 1) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

But we don't know $\rho(\pi)$!

State-action value estimation

$$Q^\pi(m, \theta_m, 1) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

Estimate it by sampling! Sample an infinite sum?

$$\sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] = \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] + o(T)$$

$$Q^\pi(m, \theta_m, 1) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

But we don't know $\rho(\pi)$!

State-action value estimation

$$Q^\pi(m, \theta_m, 1) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

Estimate it by sampling! Sample an infinite sum?

$$\sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] = \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0, a_0\right] + o(T)$$

$$Q^\pi(m, \theta_m, 1) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi) | s_0 = (m, \theta_m), a_0 = 1\right]$$

But we don't know $\rho(\pi)$!

State-action value estimation

Look at the *difference*:

$$Q^{\pi}(m,\theta_m,1) - Q^{\pi}(m,\theta_m,0) \sim \sum_{t=0}^{T}\mathbb{E}\left[r_t^{\pi} - \rho(\pi)|(m,\theta_m),1\right] - \sum_{t=0}^{T}\mathbb{E}\left[r_t^{\pi} - \rho(\pi)|(m,\theta_m),0\right]$$

$$\sum_{t=0}^{T}\mathbb{E}\left[r_t^{\pi}|(m,\theta_m),1\right] - \sum_{t=0}^{T}\mathbb{E}\left[r_t^{\pi}|(m,\theta_m),0\right]$$

$$Q^{\pi}(m,\theta_m,1) - Q^{\pi}(m,\theta_m,0) = \sum_{t=0}^{T}\mathbb{E}\left[r_t^{\pi}|(m,\theta_m),1\right] - \sum_{t=0}^{T}\mathbb{E}\left[r_t^{\pi}|(m,\theta_m),0\right]$$

Do we have to repeatedly modify the current policy?

State-action value estimation

Look at the *difference*:

$$Q^\pi(m,\theta_m,1) - Q^\pi(m,\theta_m,0) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi)|(m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi)|(m,\theta_m),0\right]$$

$$\sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),0\right]$$

$$Q^\pi(m,\theta_m,1) - Q^\pi(m,\theta_m,0) = \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),0\right]$$

Do we have to repeatedly modify the current policy?

State-action value estimation

Look at the *difference*:

$$Q^\pi(m,\theta_m,1) - Q^\pi(m,\theta_m,0) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi)|(m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi - \rho(\pi)|(m,\theta_m),0\right]$$

$$\sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),0\right]$$

$$Q^\pi(m,\theta_m,1) - Q^\pi(m,\theta_m,0) = \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^\pi|(m,\theta_m),0\right]$$

Do we have to repeatedly modify the current policy?

State-action value estimation

Look at the *difference*:

$$Q^{\pi}(m,\theta_m,1) - Q^{\pi}(m,\theta_m,0) \sim \sum_{t=0}^{T} \mathbb{E}\left[r_t^{\pi} - \rho(\pi) | (m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^{\pi} - \rho(\pi) | (m,\theta_m),0\right]$$

$$\sum_{t=0}^{T} \mathbb{E}\left[r_t^{\pi} | (m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^{\pi} | (m,\theta_m),0\right]$$

$$Q^{\pi}(m,\theta_m,1) - Q^{\pi}(m,\theta_m,0) = \sum_{t=0}^{T} \mathbb{E}\left[r_t^{\pi} | (m,\theta_m),1\right] - \sum_{t=0}^{T} \mathbb{E}\left[r_t^{\pi} | (m,\theta_m),0\right]$$

Do we have to repeatedly modify the current policy?

Gradient estimation

State-action value estimation.

Lemma

- $Q^\pi(m, d_1, 0) = Q^\pi(m, d_2, 0)$
- $Q^\pi(m, d_1, 1) = Q^\pi(m, d_2, 1) + r_{d_1} - r_{d_2}$

Proof.

$$Q^\pi(m, d, a) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | (m, d), a\right] = a \cdot r_d + \sum_{t=1}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | m, a\right]$$

□

We can sample the (sign of the) gradient in a smart way!

We do not have to repeatedly modify the current policy

Gradient estimation

State-action value estimation.

Lemma

- $Q^\pi(m, d_1, 0) = Q^\pi(m, d_2, 0)$
- $Q^\pi(m, d_1, 1) = Q^\pi(m, d_2, 1) + r_{d_1} - r_{d_2}$

Proof.

$$Q^\pi(m, d, a) = \sum_{t=0}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | (m, d), a\right] = a \cdot r_d + \sum_{t=1}^{\infty} \mathbb{E}\left[r_t^\pi - \rho(\pi) | m, a\right]$$

□

We can sample the (sign of the) gradient in a smart way!

We do not have to repeatedly modify the current policy

Polynomial complexity in M

Theorem

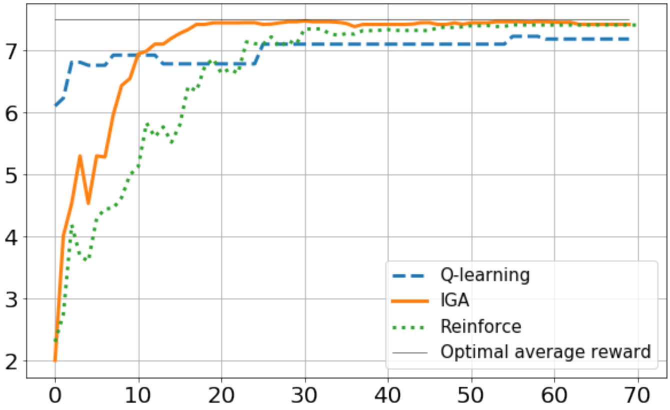
- *M memory slots*
- *N flow categories*
- $\overline{Q} := \min\{|Q^\pi(m,i,1) - Q^\pi(m,i,0)| \neq 0\}$.

IGA converges with probability $1-\delta$ to an optimal policy in a number of steps

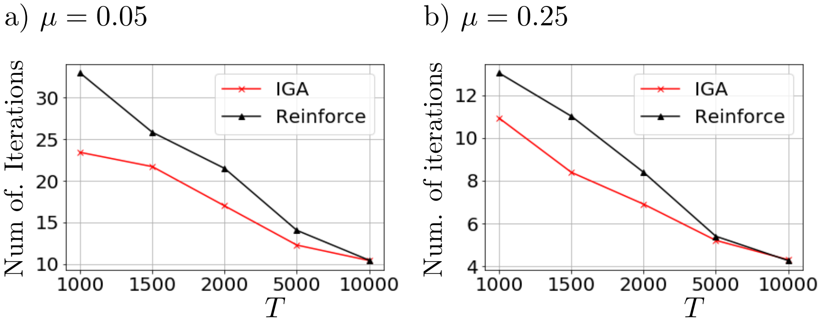
$$O\left(\frac{M^N \ln(\varepsilon_1^{-1}) \ln(\delta^{-1})}{\varepsilon_2^2}\right).$$

where $\varepsilon_1 + \varepsilon_2 < \frac{1}{2}\overline{Q}$.

Experiments: convergence (1/3)



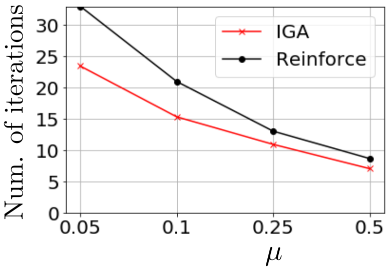
Experiments: convergence (2/3)



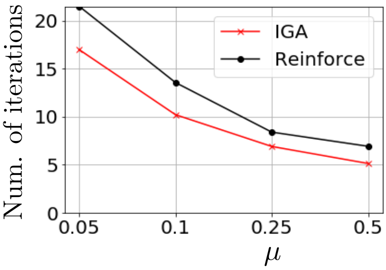
Iterations to 98% optimality: increasing episode length T. $D = 4$, $r = \{20, 15, 4, 3\}$, $\lambda = \{0.1, 0.2, 0.4, 0.3\}$

Experiments: convergence (3/3)

a) $T = 1000$



b) $T = 2000$



Iterations to 98% optimality: increasing service rate. $D = 4$, $r = \{20, 15, 4, 3\}$, $\lambda = \{0.1, 0.2, 0.4, 0.3\}$

Conclusions

- **Optimal Trunk Reservation:** optimal admission control has a K-threshold structure
- **Learning Online:** no need to known arrival rates and the departure rates
- **Convergence:** using discrete step proves much faster than legacy approaches (Q-learning, Reinforce)

M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

B. L. Miller, "A queueing reward system with several customer classes," *Management Science*, vol. 16, no. 3, pp. 234–245, 1969.

R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press Cambridge, 1998, vol. 1, no. 1.

Gradient estimation. Naive algorithm.

- Generate an episode of length N
- Choose a sampling window T
- Sample state-action values by a sliding window

$$(\underbrace{(m_1, d_1, r_1, a_1), \dots, (m_T, d_T, r_T, a_T)}_{\sum_{j=1}^{T} a_i d_i}, (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \dots, (m_N, d_N, r_N, a_N))$$

$$Q(m_1, d_1, a_1) = Q(m_1, d_1, a_1) + \sum_{j=1}^{T} a_i r_i$$

Gradient estimation. Naive algorithm.

- Generate an episode of length N
- Choose a sampling window T
- Sample state-action values by a sliding window

$$\underbrace{((m_1, d_1, r_1, a_1), \ldots, (m_T, d_T, r_T, a_T)}_{\sum_{j=1}^{T} a_i d_i}, (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \ldots, (m_N, d_N, r_N, a_N))$$

$$Q(m_1, d_1, a_1) = Q(m_1, d_1, a_1) + \sum_{j=1}^{T} a_i r_i$$

Gradient estimation. Naive algorithm.

- Generate an episode of length N
- Choose a sampling window T
- Sample state-action values by a sliding window

$$((m_1, d_1, r_1, a_1), \underbrace{..., (m_T, d_T, r_T, a_T), (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1})}_{\sum_{j=2}^{T+1} a_i d_i}, ..., (m_N, d_N, r_N, a_N))$$

$$Q(m_2, d_2, a_2) = Q(m_2, d_2, a_2) + \sum_{j=2}^{T+2} a_i r_i$$

Gradient estimation. Naive algorithm.

- Generate an episode of length N
- Choose a sampling window T
- Sample state-action values by a sliding window

$$((m_1, d_1, r_1, a_1), \ldots, (m_T, d_T, r_T, a_T), (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \underbrace{\ldots, (m_N, d_N, r_N, a_N)}_{\sum_{j=N-T}^{N} a_i r_i})$$

$$Q(m_{N-T}, d_{N-T}, a_{N-T}) = Q(m_{N-T}, d_{N-T}, a_{N-T}) + \sum_{j=N-T}^{N} a_i r_i$$

And average

Gradient estimation. Smart algorithm.

- Generate an episode of length N
- Choose a sampling window T
- Sample state-action values by a sliding window
- Pretend: $(m, d, 1) \to (m, \theta_m, 1), (m, d, 0) \to (m, \theta_m, 0)$

$$((m_1, d_1, r_1, a_1), \ldots, \underbrace{(m_i, d_i, r_i, a_i), \ldots, (m_{i+T}, d_{i+T}, r_{i+T}, a_{i+T})}_{a_i r_{\theta_{m_i}} + \sum_{j=i+1}^{i+T} a_i r_i} \ldots, (m_N, d_N, r_N, a_N))$$

$$Q(m_i, \theta_{m_i}, 1) = Q(m_i, \theta_{m_i}, 1) + a_i r_{\theta_{m_i}} + \sum_{j=i+1}^{i+T} a_i r_i, \text{ if } a_i = 1$$

$$Q(m_i, \theta_{m_i}, 0) = Q(m_i, \theta_{m_i}, 0) + 0 + \sum_{j=i+1}^{i+T} a_i r_i, \text{ if } a_i = 0$$

And average

Gradient estimation algorithm

Algorithm 1: $\texttt{LearnGradient}(\pi, T, N)$

input: $\pi = (\theta_0, \ldots, \theta_{M-1}) \in \{0, \ldots, D\}^M, T, N \in \mathbb{N}$
compute: $r_m^\pi = r_{\theta_m}, m = 0, \ldots, M-1$
initialize: $Q_0 = [[], \ldots, []]$, long M
$Q_1 = [[], \ldots, []]$, long M
generate episode $\{(m_i, d_i, a_i)\}_{i=0,\ldots,N}$ according to current policy π
for $i = 0, \ldots, N-T$ **do**
  if $a_i == 1$ **then**
    $q_1 = r_{m_i}^\pi + \sum_{t=i+1}^{T} r_t \cdot a_t$
    $Q_1[m_i].append(q_1)$
  else
    $q_0 = \sum_{t=i+1}^{T} r_t \cdot a_t$
    $Q_0[m_i].append(q_0)$
  end if
end for
for $m = 0, \ldots, M-1$ **do**
  $\widehat{Q}_m = average(Q_1[m]) - average(Q_0[m])$
end for
return $\widehat{Q}_0, \ldots, \widehat{Q}_{M-1}$

Definition of a flow

An IP traffic flow is a set of packets identified by the following attributes:

- IP source address
- IP destination address
- Source port
- Destination port
- Layer 3 protocol type
- Class of Service
- Router or switch interface

Markov Decision Problem formulation: transition probabilities

$$P = \begin{array}{|c|c|c|c|} \hline p_1 & p_2 & \dots & p_n \\ \hline p_1 & p_2 & \dots & p_n \\ \hline \dots & \dots & \dots & \dots \\ \hline p_1 & p_2 & \dots & p_n \\ \hline \end{array}$$

$a = 0$

P	0	0	0
λP	$(1-\lambda)P$	0	0
$\lambda^2 P$	$\binom{2}{1}\lambda(1-\lambda)P$	$(1-\lambda)^2 P$	0
$\lambda^3 P$	$\binom{3}{1}\lambda^2(1-\lambda)P$	$\binom{3}{2}\lambda(1-\lambda)^2 P$	$(1-\lambda)^3 P$

$a = 1$

λP	$(1-\lambda)P$	0	0
$\lambda^2 P$	$\binom{2}{1}\lambda(1-\lambda)P$	$(1-\lambda)^2 P$	0
$\lambda^3 P$	$\binom{3}{1}\lambda^2(1-\lambda)P$	$\binom{3}{2}\lambda(1-\lambda)^2 P$	$(1-\lambda)^3 P$