Optimal Trunk Reservation by Policy Learning

\textit{Antonio Massaro} \textsuperscript{1}     \textit{Francesco De Pellegrini} \textsuperscript{1, 2}     \textit{Lorenzo Maggi} \textsuperscript{3}

\textsuperscript{1} Nokia Bell Labs, Paris. Fondazione Bruno Kessler, Trento, Italy
\textsuperscript{2} University of Avignon, France
\textsuperscript{3} Nokia Bell Labs, Paris

12/04/2019

\textsuperscript{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: $\lambda_1, \ldots, \lambda_D$
- Poisson service time $\mu$
- States: $(m, d) \in \{0, \ldots, M\} \times \{1, \ldots, D\}$
- Actions: $A(m, d) = \{0, 1\}$, $A(M, d) = \{0\}$
- Rewards: $r_1 > r_2 > \ldots > r_D$.
- Policy: $\pi_t(m, d) = \mathbb{P}(a(m, d) = 1)$

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

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:

∃ 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
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

![Diagram showing buffer occupation by job class](image-url)
Policy search: basic idea

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

0

1

![Graphs showing buffer occupation over job classes for states 0 and 1]
Policy search: basic idea

Idea: progressively ‘fill’ the probability of admission at each state
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
Policy search: basic idea

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

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

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

\[ (\theta_1, ..., \theta_M) \in [0, D]^M \]
A differentiable parametrization for policies

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

\[ (\theta_1, \ldots, \theta_M) \in [0, D]^M \]
A differentiable parametrization of the average reward

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

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

\[ \theta_1, \ldots, \theta_M \mapsto \pi_\theta \mapsto \rho(\pi_\theta) \]

Take the gradient and do gradient ascent!
A differentiable parametrization of the average reward

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

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

(\theta_1, \ldots, \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 \( \pi \)
- \( Q^\pi(s, a) = \sum_{t=0}^{\infty} E[r^\pi_t - \rho(\pi)|s_0 = s, a_0 = a] \)

In our case it is very simple!

\[ \frac{\partial \rho}{\partial \theta_m} = p^\pi(m, [\theta_m])(Q^\pi(m, [\theta_m], 1) - Q^\pi(m, [\theta_m], 0)). \]
A formula for the gradient of the average reward\cite{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 \(\pi\)
- \(Q^\pi(s, a) = \sum_{t=0}^{\infty} \mathbb{E}[r_t^\pi - \rho(\pi)|s_0 = s, a_0 = a]\)

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

\[ \pi = (3.5, 2.5, 2.5, 2.5, 1.5, 1.5, .5, .5) \]

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

Gradient sign
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, $\pi$ satisfies Bellman equation, hence it is optimal

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, $\pi$ satisfies Bellman equation, hence it is optimal

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

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

Estimate it by sampling! Sample an infinite sum?

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

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

But we don’t know \( \rho(\pi) \)!
State-action value estimation

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

Estimate it by sampling! Sample an infinite sum?

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

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

But we don’t know \( \rho(\pi) \)!
State-action value estimation

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

Estimate it by sampling! Sample an infinite sum?

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

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

But we don’t know \( \rho(\pi) \)!
State-action value estimation

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

Estimate it by sampling! Sample an infinite sum?

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

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

But we don’t know \(\rho(\pi)!\)
State-action value estimation

Look at the difference:

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

Do we have to repeatedly modify the current policy?
State-action value estimation

Look at the difference:

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

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

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

Do we have to repeatedly modify the current policy?
State-action value estimation

Look at the difference:

\[ Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0) \sim \sum_{t=0}^{T} \mathbb{E}[r^\pi_t - \rho(\pi)(m, \theta_m), 1] - \sum_{t=0}^{T} \mathbb{E}[r^\pi_t - \rho(\pi)(m, \theta_m), 0] \]

\[ \sum_{t=0}^{T} \mathbb{E}[r^\pi_t (m, \theta_m), 1] - \sum_{t=0}^{T} \mathbb{E}[r^\pi_t (m, \theta_m), 0] \]

\[ Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0) = \sum_{t=0}^{T} \mathbb{E}[r^\pi_t | (m, \theta_m), 1] - \sum_{t=0}^{T} \mathbb{E}[r^\pi_t | (m, \theta_m), 0] \]

Do we have to repeatedly modify the current policy?
State-action value estimation

Look at the difference:

\[ Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0) \sim \sum_{t=0}^{T} \mathbb{E} [r^\pi_t - \rho(\pi)(m, \theta_m), 1] - \sum_{t=0}^{T} \mathbb{E} [r^\pi_t - \rho(\pi)(m, \theta_m), 0] \]

\[ \sum_{t=0}^{T} \mathbb{E} [r^\pi_t | (m, \theta_m), 1] - \sum_{t=0}^{T} \mathbb{E} [r^\pi_t | (m, \theta_m), 0] \]

\[ Q^\pi(m, \theta_m, 1) - Q^\pi(m, \theta_m, 0) = \sum_{t=0}^{T} \mathbb{E} [r^\pi_t | (m, \theta_m), 1] - \sum_{t=0}^{T} \mathbb{E} [r^\pi_t | (m, \theta_m), 0] \]

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.

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

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.

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

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)

a) $\mu = 0.05$

b) $\mu = 0.25$

**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)


Gradient estimation. Naive algorithm.

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

$$\left((m_1, d_1, r_1, a_1), \ldots, (m_T, d_T, r_T, a_T), (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \ldots, (m_N, d_N, r_N, a_N)\right)$$

$$\sum_{j=1}^{T} a_i d_i$$

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

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

\[
((m_1, d_1, r_1, a_1), \ldots, (m_T, d_T, r_T, a_T), (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \ldots, (m_N, d_N, r_N, a_N))
\]

\[
Q(m_1, d_1, a_1) = Q(m_1, d_1, a_1) + \sum_{j=1}^{T} a_i r_i
\]
Gradient estimation. Naive algorithm.

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

$$((m_1, d_1, r_1, a_1), \ldots, (m_T, d_T, r_T, a_T), (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \ldots, (m_N, d_N, r_N, a_N))$$

$$\sum_{j=2}^{T+1} a_i d_i$$

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

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

$$((m_1, d_1, r_1, a_1), \ldots, (m_T, d_T, r_T, a_T), (m_{T+1}, d_{T+1}, r_{T+1}, a_{T+1}), \ldots, (m_N, d_N, r_N, a_N))$$

$$\sum_{j=N-T}^{N} a_i r_i$$

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

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: $(m, d, 1) \rightarrow (m, \theta_m, 1), (m, d, 0) \rightarrow (m, \theta_m, 0)$

\[
\begin{align*}
\left((m_1, d_1, r_1, a_1), \ldots, (m_i, d_i, r_i, a_i), \ldots, (m_i+T, d_i+T, r_i+T, a_i+T) \ldots, (m_N, d_N, r_N, a_N)\right) \\
a_i r_{\theta_m} + \sum_{j=i+1}^{i+T} a_i r_i
\end{align*}
\]

\[
\begin{align*}
Q(m_i, \theta_{m_i}, 1) &= Q(m_i, \theta_{m_i}, 1) + a_i r_{\theta_{m_i}} + \sum_{j=i+1}^{i+T} a_i r_i, \text{ if } a_i = 1 \\
Q(m_i, \theta_{m_i}, 0) &= Q(m_i, \theta_{m_i}, 0) + 0 + \sum_{j=i+1}^{i+T} a_i r_i, \text{ if } a_i = 0
\end{align*}
\]

And average
Algorithm 1: LearnGradient(\(\pi, T, N\))

\begin{verbatim}
input: \(\pi = (\theta_0, \ldots, \theta_{M-1}) \in \{0, \ldots, D\}^M, T, N \in \mathbb{N}\)
compute: \(r^\pi_m = r_{\theta_m}, m = 0, \ldots, M - 1\)
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 \(\pi\)
for \(i = 0, \ldots, N - T\) do
  if \(a_i == 1\) then
    \(q_1 = r^\pi_{m_i} + \sum_{t=i+1}^{T} r_t \cdot a_t\)
    \(Q_1[m_i].append(q_1)\)
  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
  \(\hat{Q}_m = \text{average}(Q_1[m]) - \text{average}(Q_0[m])\)
end for
return \(\hat{Q}_0, \ldots, \hat{Q}_{M-1}\)
\end{verbatim}
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

$$P = \begin{bmatrix}
p_1 & p_2 & \ldots & p_n \\
p_1 & p_2 & \ldots & p_n \\
\vdots & \vdots & \ddots & \vdots \\
p_1 & p_2 & \ldots & p_n 
\end{bmatrix}$$

\(a = 0\)

\[
\begin{array}{c|c|c|c}
   P & 0 & 0 & 0 \\
   \lambda P & (1 - \lambda)P & 0 & 0 \\
   \lambda^2 P & \binom{2}{1} \lambda(1 - \lambda)P & (1 - \lambda)^2P & 0 \\
   \lambda^3 P & \binom{3}{1} \lambda^2(1 - \lambda)P & \binom{3}{2} \lambda(1 - \lambda)^2P & (1 - \lambda)^3P \\
\end{array}
\]

\(a = 1\)

\[
\begin{array}{c|c|c|c}
   \lambda P & (1 - \lambda)P & 0 & 0 \\
   \lambda^2 P & \binom{2}{1} \lambda(1 - \lambda)P & (1 - \lambda)^2P & 0 \\
   \lambda^3 P & \binom{3}{1} \lambda^2(1 - \lambda)P & \binom{3}{2} \lambda(1 - \lambda)^2P & (1 - \lambda)^3P \\
\end{array}
\]