Score-Aware Policy-Gradient Methods and Performance Guarantees using Local Lyapunov Conditions

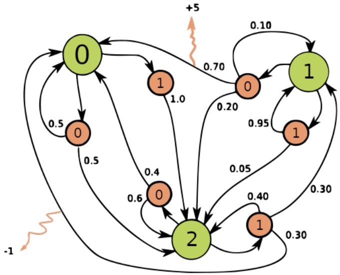
Céline Comte[1], Matthieu Jonckheere[1], Jaron Sanders[2], and Albert Senen-Cerda[1,2]

[1]CNRS, LAAS, and IRIT, Toulouse, France
[2]Eindhoven University of Technology, Eindhoven, The Netherlands

February 16, 2024 – IMT Atlantique

Reinforcement learning

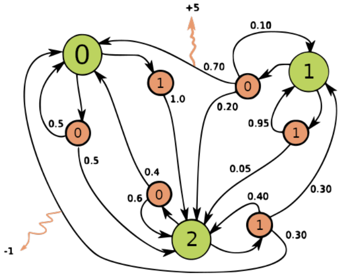
- **Markov decision process** (MDP)



Source: Wikipedia (modified)

Reinforcement learning

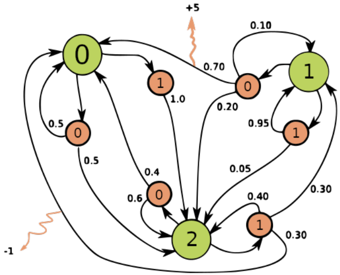
- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$



Source: Wikipedia (modified)

Reinforcement learning

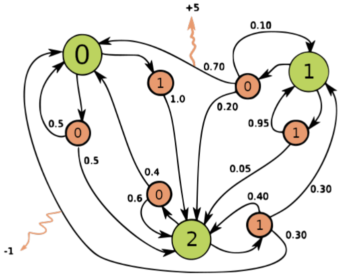
- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$
  - Policy $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$ parameterized by θ



Source: Wikipedia (modified)

Reinforcement learning

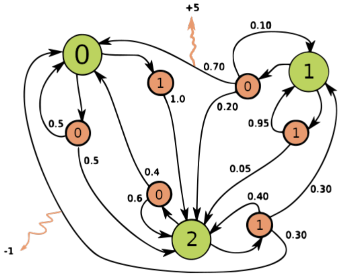
- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$
  - Policy $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$ parameterized by θ
  - Environment $P(r, s'|s,a) = \mathbb{P}\left[\begin{smallmatrix} R_{t+1}=r \\ S_{t+1}=s' \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$



Source: Wikipedia (modified)

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$
  - Policy $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$ parameterized by θ
  - Environment $P(r, s'|s,a) = \mathbb{P}\left[\begin{smallmatrix} R_{t+1}=r \\ S_{t+1}=s' \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
- **Goal:** Find a θ that maximizes the **average reward rate**

$$J(\theta) = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t]$$

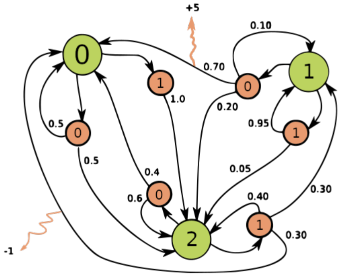


Source: Wikipedia (modified)

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Policy $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$ parameterized by θ
  - Environment $P(r, s'|s,a) = \mathbb{P}\left[\begin{smallmatrix} R_{t+1}=r \\ S_{t+1}=s' \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
- **Goal:** Find a θ that maximizes the **average reward rate**

$$J(\theta) = \lim_{T\to+\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t]$$

$$= \sum_s p(s|\theta) \sum_a \pi(a|s,\theta) \sum_{s',r} r P(r,s'|s,a)$$



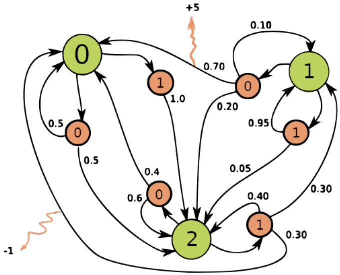
Source: Wikipedia (modified)

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Policy $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$ parameterized by θ
  - Environment $P(r, s'|s,a) = \mathbb{P}\left[\begin{smallmatrix} R_{t+1}=r \\ S_{t+1}=s' \end{smallmatrix} \,\middle|\, \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
- **Goal:** Find a θ that maximizes the **average reward rate**

$$J(\theta) = \lim_{T\to+\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t]$$

$$= \sum_s p(s|\theta) \sum_a \pi(a|s,\theta) \sum_{s',r} r P(r,s'|s,a)$$

Stationary distribution of $(S_t, t \ge 0)$ under $\pi(a|s,\theta)$



Source: Wikipedia (modified)

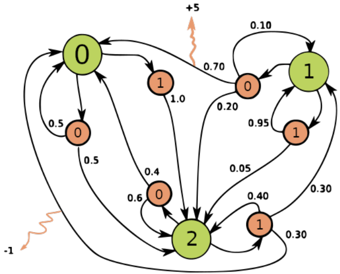
Reinforcement learning

- **Markov decision process** (MDP) with
  - State-action-reward sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$
  - Policy $\pi(a|s,\theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$ parameterized by θ
  - Environment $P(r,s'|s,a) = \mathbb{P}\left[\begin{smallmatrix} R_{t+1}=r \\ S_{t+1}=s' \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
- **Goal:** Find a θ that maximizes the **average reward rate**

$$J(\theta) = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[R_t]$$

$$= \sum_{s} p(s|\theta) \sum_{a} \pi(a|s,\theta) \sum_{s',r} r P(r,s'|s,a)$$

- We estimate the **policy gradient** $\nabla J(\theta)$ and apply **stochastic gradient ascent**



Source: Wikipedia (modified)

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:

1: Initialize S_0 and Θ_0
2: **for** $t = 0, 1, 2, \dots$ **do**
3: Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
4: Take action A_t and observe R_{t+1}, S_{t+1}
5: Estimate $\nabla J(\Theta_t)$ using the history $S_0, \Theta_0, A_0, R_1, \dots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$
6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha \nabla J(\Theta_t)$
7: **end for**

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:

Initialize S_0 and Θ_0
for $t = 0, 1, 2, \ldots$ **do**
    Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
    Take action A_t and observe R_{t+1}, S_{t+1}
    Estimate $\nabla J(\Theta_t)$ using the history $S_0, \Theta_0, A_0, R_1, \ldots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
    Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha \nabla J(\Theta_t)$
end for

Policy-gradient algorithms

- Typical **policy-gradient algorithm**:
  1: Initialize S_0 and Θ_0
  2: **for** $t = 0, 1, 2, \dots$ **do**
  3: Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
  4: Take action A_t and observe R_{t+1}, S_{t+1}
  5: Estimate $\nabla J(\Theta_t)$ using the history $S_0, \Theta_0, A_0, R_1, \dots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
  6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha \nabla J(\Theta_t)$
  7: **end for**
- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$\nabla J(\theta) \propto \mathbb{E}\Big[\Big(R - J(\theta) + v(S'|\theta) - v(S|\theta)\Big)\nabla \log \pi(A|S, \theta)\Big],$$

with $(S, A, R, S') \sim \textsc{stat}(\theta)$: $\mathbb{P}[S = s, A = a, R = r, S' = s'] = p(s|\theta)\pi(a|s, \theta)P(r, s'|s, a)$.

Our approach

- Consider MDPs and policies $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$

Our approach

- Consider MDPs and policies $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
  Examples: M/M/1 queue, Jackson network, Ising model, load-balancing systems, multi-server queues with redundancy scheduling, online matching queues . . .

Our approach

- Consider MDPs and policies $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
  Examples: M/M/1 queue, Jackson network, Ising model, load-balancing systems, multi-server queues with redundancy scheduling, online matching queues ...
- Exploit the product-form to introduce a new **policy-gradient estimator**

Our approach

- Consider MDPs and policies $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
  Examples: M/M/1 queue, Jackson network, Ising model, load-balancing systems, multi-server queues with redundancy scheduling, online matching queues ...
- Exploit the product-form to introduce a new **policy-gradient estimator**
- Prove that the corresponding policy-gradient algorithm has nice **convergence properties**

Our approach

- Consider MDPs and policies $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t \geq 0)$ has a **product-form stationary distribution** $p(s|\theta)$
  Examples: M/M/1 queue, Jackson network, Ising model, load-balancing systems, multi-server queues with redundancy scheduling, online matching queues ...
- Exploit the product-form to introduce a new **policy-gradient estimator**
- Prove that the corresponding policy-gradient algorithm has nice **convergence properties**
- Outline of the rest of the presentation:
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \boxed{\rho_i(\theta)}^{x_i(s)}$$

Depends on θ

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

Depends on s

Depends on θ

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Partition function** Z

$$Z(\theta) = \sum_{s} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Partition function** Z

$$Z(\theta) = \sum_{s} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$ and **feature function** $x = (x_1, x_2, \ldots, x_n)$

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

Take the log →

Exponential family of distributions

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

- **Partition function** Z

$$Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$ and **feature function** $x = (x_1, x_2, \ldots, x_n)$

① Product-form distributions as exponential families

- **Product-form** distribution

$$p(s|\theta) = \frac{1}{Z(\theta)} \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

Take the log →

Exponential family of distributions

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

- **Partition function** Z

$$Z(\theta) = \sum_s \prod_{i=1}^{n} \rho_i(\theta)^{x_i(s)}$$

Take the log →

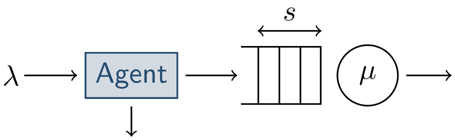
Log-partition function

$$\log Z(\theta) = \log \left(\sum_s e^{\log \rho(\theta)^\intercal x(s)} \right)$$

- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$ and **feature function** $x = (x_1, x_2, \dots, x_n)$

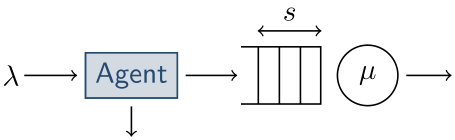
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \ldots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



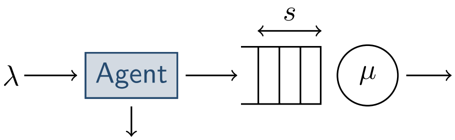
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \ldots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{admit}|s, \theta) = \dfrac{1}{1 + e^{-\theta_s}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \ldots)$



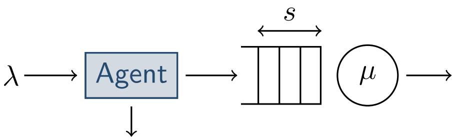
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{admit}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$



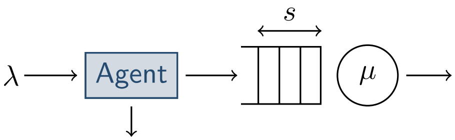
Example: M/M/1 queue with admission control

- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{admit}|s, \theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{admit}|s, \theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta)s \right) \times \dfrac{1}{\lambda}$

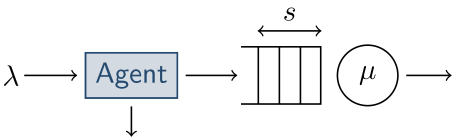
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{admit}|s, \theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{admit}|s, \theta) \right)}_{\text{Probability of accepting a job}} - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$

Example: M/M/1 queue with admission control

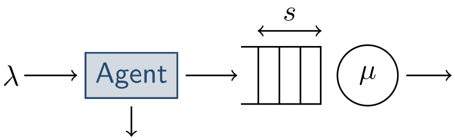
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{admit}|s, \theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{admit}|s, \theta) \right)}_{\text{Probability of accepting a job}} - \eta \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) s \right)}_{\text{Mean queue size}} \times \dfrac{1}{\lambda}$

Example: M/M/1 queue with admission control

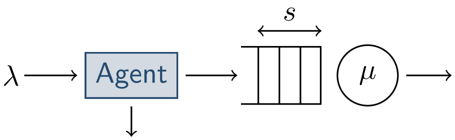
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{admit}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{admit}|s,\theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left(\dfrac{\lambda}{\mu} \pi(\text{admit}|i,\theta) \right)^{1_{\{s \geq i\}}} \left(\dfrac{\lambda}{\mu} \pi(\text{admit}|k,\theta) \right)^{\max(s-k,0)}$

Example: M/M/1 queue with admission control

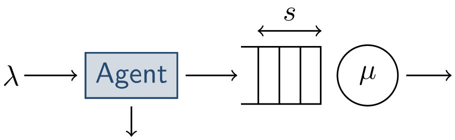
- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit



- Policy $\pi(\text{admit}|s, \theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{admit}|s, \theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left(\frac{\lambda}{\mu} \pi(\text{admit}|i, \theta) \right)^{1_{\{s \geq i\}}} \left(\frac{\lambda}{\mu} \pi(\text{admit}|k, \theta) \right)^{\max(s-k, 0)}$

Depends on θ

Example: M/M/1 queue with admission control



- Arrival rate $\lambda > 0$, service rate $\mu > 0$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Admission reward α per job
- Holding cost rate η per job per time unit
- Policy $\pi(\text{admit}|s,\theta) = \dfrac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter $\theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_k)$
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta)\pi(\text{admit}|s,\theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \dfrac{1}{\lambda}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=0}^{k-1} \left(\frac{\lambda}{\mu} \pi(\text{admit}|i,\theta) \right)^{1_{\{s \geq i\}}} \left(\frac{\lambda}{\mu} \pi(\text{admit}|k,\theta) \right)^{\max(s-k,0)}$

Depends on s

Depends on θ

② Score-aware gradient estimator (SAGE)

- The **score** is the gradient of the log-likelihood with respect to the parameter vector:

$$\text{“Likelihood”} = p(s|\theta) \quad \to \quad \text{“Score”} = \nabla \log p(s|\theta).$$

② Score-aware gradient estimator (SAGE)

- The **score** is the gradient of the log-likelihood with respect to the parameter vector:

$$\text{“Likelihood”} = p(s|\theta) \quad \to \quad \text{“Score”} = \nabla \log p(s|\theta).$$

Theorem

If $p(s|\theta)$ *has a product-form in the sense that* $\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$, *then*

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)]),$$
$$\nabla J(\theta) = \mathrm{D} \log \rho(\theta)^\intercal \mathrm{Cov}[R, x(S)] + \mathbb{E}[R \nabla \log \pi(A|S,\theta)],$$

with $(S, A, R, S') \sim \text{STAT}(\theta) : \mathbb{P}[S = s, A = a, R = r, S' = s'] = p(s|\theta)\pi(a|s,\theta)P(r, s'|s,a)$.

② Score-aware gradient estimator (SAGE)

- The **score** is the gradient of the log-likelihood with respect to the parameter vector:

$$\text{“Likelihood”} = p(s|\theta) \quad \rightarrow \quad \text{“Score”} = \nabla \log p(s|\theta).$$

Theorem

If $p(s|\theta)$ *has a product-form in the sense that* $\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$, *then*

$$\begin{aligned} \nabla \log p(s|\theta) &= \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)]), \\ \nabla J(\theta) &= \mathrm{D} \log \rho(\theta)^\intercal \mathrm{Cov}[R, x(S)] + \mathbb{E}[R \nabla \log \pi(A|S,\theta)], \end{aligned}$$

with $(S, A, R, S') \sim \textsc{stat}(\theta) : \mathbb{P}[S = s, A = a, R = r, S' = s'] = p(s|\theta)\pi(a|s,\theta)P(r,s'|s,a)$.

- **Main take-away:** If we can compute $\mathrm{D} \log \rho(\theta)$, we have an estimator for $\nabla J(\theta)$.

③ SAGE-based policy-gradient algorithm

- Typical **policy-gradient algorithm**:

1: Initialize S_0 and Θ_0
2: **for** $t = 0, 1, 2, \dots$ **do**
3: Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
4: Take action A_t and observe R_{t+1}, S_{t+1}
5: Estimate $\nabla J(\Theta_t)$ using the history $S_0, \Theta_0, A_0, R_1, \dots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha \nabla J(\Theta_t)$
7: **end for**

③ SAGE-based policy-gradient algorithm

- Typical **policy-gradient algorithm**:

1: Initialize S_0 and Θ_0
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Sample $A_t \sim \pi(\cdot|S_t, \Theta_t)$
4: Take action A_t and observe R_{t+1}, S_{t+1}
5: Estimate $\nabla J(\Theta_t)$ using the history $S_0, \Theta_0, A_0, R_1, \ldots, S_t, \Theta_t, A_t, R_{t+1}, S_{t+1}$ How?
6: Update $\Theta_{t+1} \leftarrow \Theta_t + \alpha \nabla J(\Theta_t)$
7: **end for**

- Instead of applying actor-critic, we estimate $J(\Theta_t)$ with a **SAGE**:

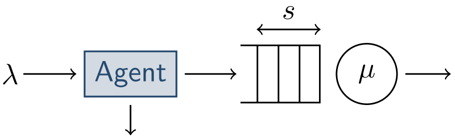
$$\nabla J(\theta) = \mathrm{D} \log \rho(\theta)^\intercal \mathrm{Cov}[R, x(S)] + \mathbb{E}[R \nabla \log \pi(A|S, \theta)],$$

with $(S, A, R, S') \sim \textsc{stat}(\theta)$: $\mathbb{P}[S = s, A = a, R = r, S' = s'] = p(s|\theta)\pi(a|s, \theta)P(r, s'|s, a)$.

Example: M/M/1 queue with admission control

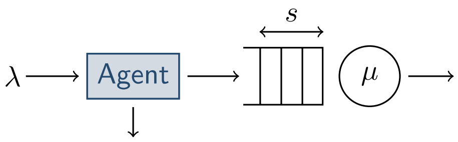
Stable/Possibly-unstable case

- Arrival rate $\lambda = 0.7/1.4$, service rate $\mu = 1$
- Admission reward $\alpha = 5$, holding cost rate $\eta = 1$
- Initial policy: admit with probability $\frac{1}{2}$



Stable/Possibly-unstable case

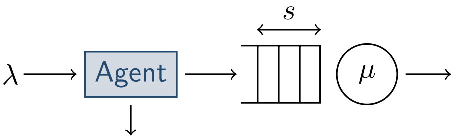
- Arrival rate $\lambda = 0.7/1.4$, service rate $\mu = 1$
- Admission reward $\alpha = 5$, holding cost rate $\eta = 1$
- Initial policy: admit with probability $\frac{1}{2}$
- Optimal policy: admit in states 0, 1, and 2, reject in states ≥ 3
  admit in states 0 and 1, reject in states ≥ 2



Example: M/M/1 queue with admission control

Stable/Possibly-unstable case

- Arrival rate $\lambda = 0.7/1.4$, service rate $\mu = 1$
- Admission reward $\alpha = 5$, holding cost rate $\eta = 1$
- Initial policy: admit with probability $\frac{1}{2}$
- Optimal policy: admit in states 0, 1, and 2, reject in states ≥ 3
  admit in states 0 and 1, reject in states ≥ 2

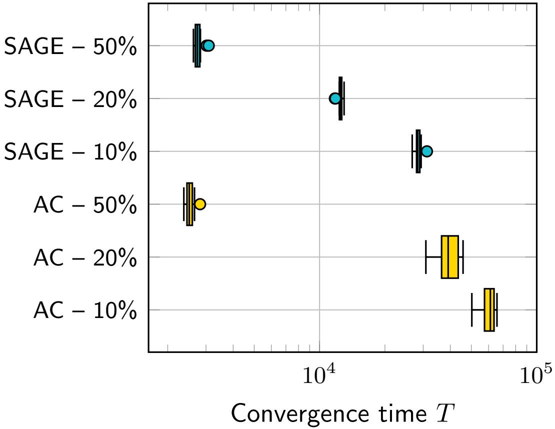


Algorithms

- SAGE-based policy-gradient
- Actor-critic without eligibility traces (Sutton and Barto, Section 13.6)
- Gradient update at every step, with step size $\alpha = 10^{-3}$

Example: M/M/1 queue with admission control

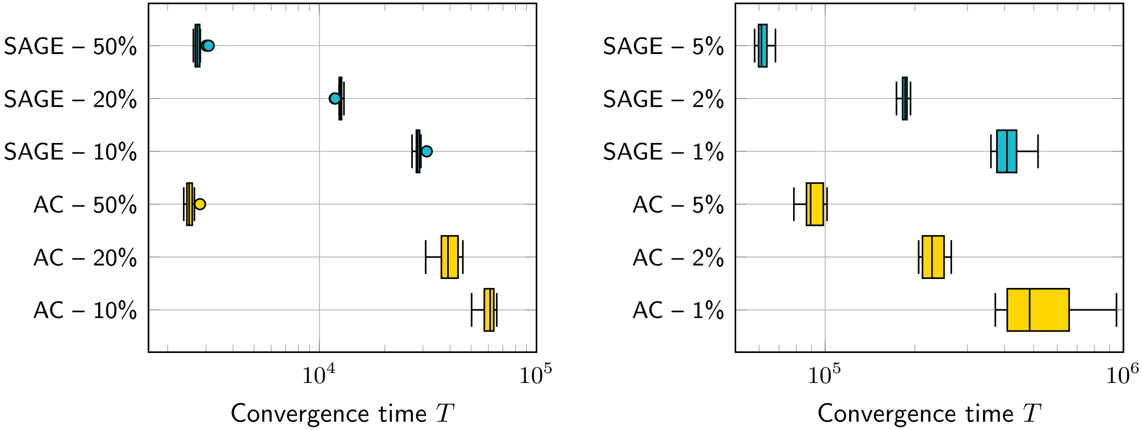
Stable case – Convergence time
(Time T such that $J(\Theta_t) > J^* - \epsilon$ for each $t \in \{T, T+1, \dots, 10^6\}$)



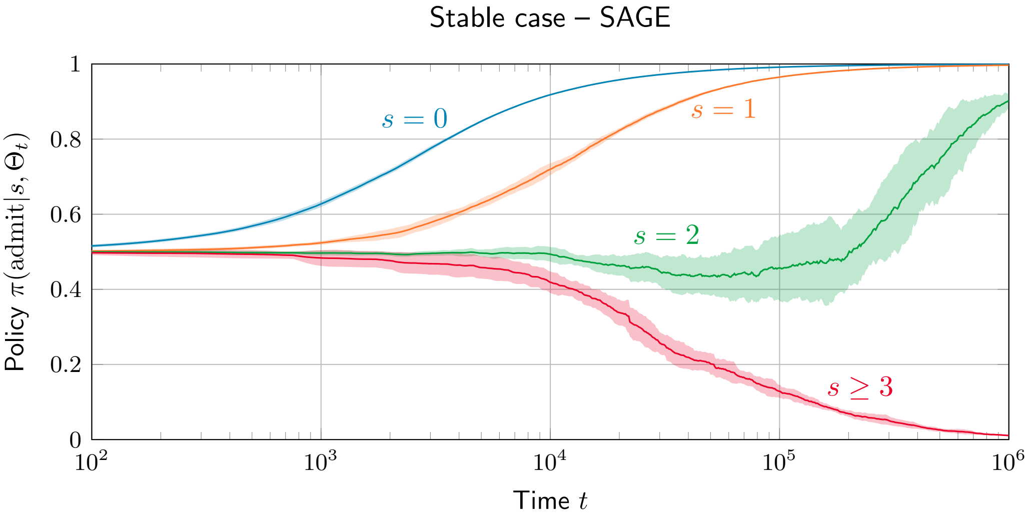
Example: M/M/1 queue with admission control

Stable case – Convergence time

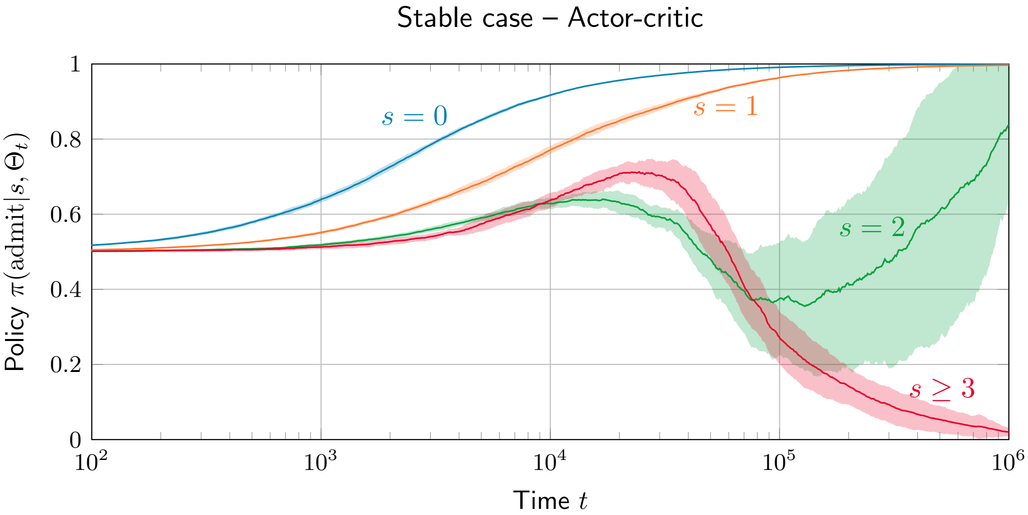
(Time T such that $J(\Theta_t) > J^* - \epsilon$ for each $t \in \{T, T+1, \ldots, 10^6\}$)



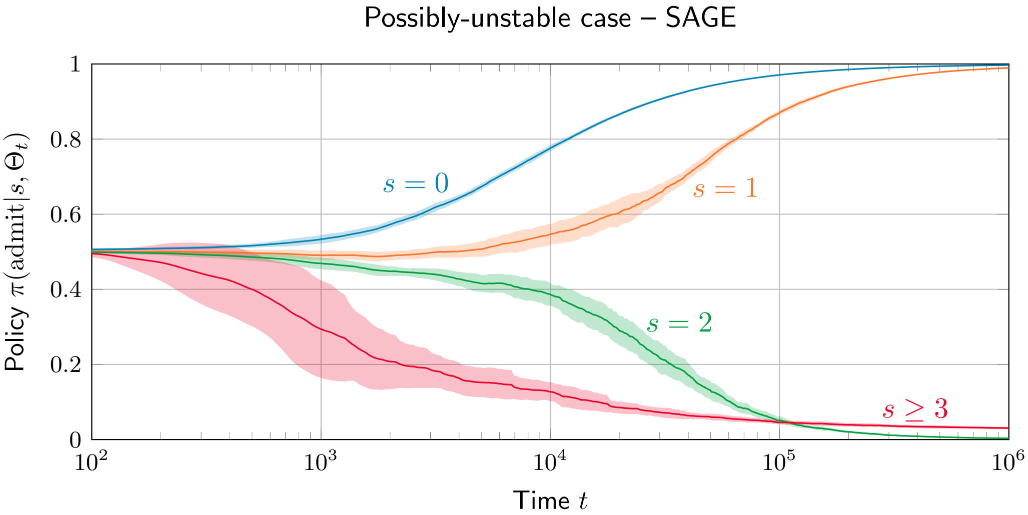
Example: M/M/1 queue with admission control



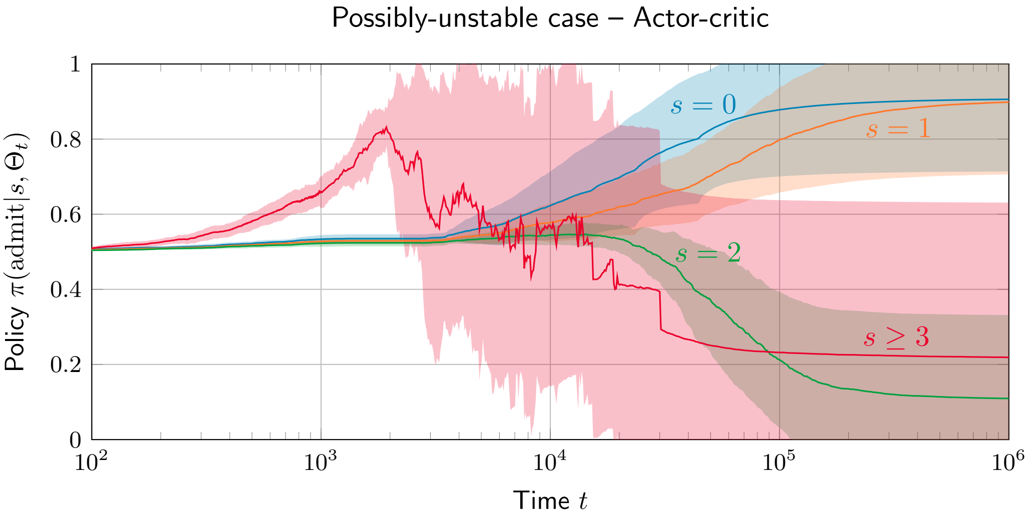
Example: M/M/1 queue with admission control



Example: M/M/1 queue with admission control



Example: M/M/1 queue with admission control



④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a <u>*batch variant*</u> *of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.*

④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will <u>converge to a global maximizer</u> with large probability.

④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint available on arXiv.

④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint available on arXiv.

What are these "additional assumptions"?

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint available on arXiv.

What are these "additional assumptions"?

- The step sizes are decreasing and the batch sizes are increasing.

④ Local convergence result

(Sketch of) Theorem

Under additional assumptions, a batch variant of the algorithm that starts in a basin of attraction of a global maximizer will converge to a global maximizer with large probability.

Proof: See preprint available on arXiv.

What are these "additional assumptions"?

- The step sizes are decreasing and the batch sizes are increasing.
- There exists a neighborhood of the global maximizer where:
  - The Markov chain of state-action pairs is geometrically ergodic.
  - The objective function behaves approximately in a convex manner in directions that are perpendicular to the set of global maximizers.
  - The function $\mathrm{D} \log \rho$ is bounded and the functions x, r, and $r \nabla \log \pi$ grow slowly enough.

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - (Ongoing) Run extensive numerical results on larger and more challenging examples.

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - (Ongoing) Run extensive numerical results on larger and more challenging examples.
  - Find better estimators for covariance and expectation, such as robust estimators.

Conclusion

- **Main contributions**
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Nonconvex convergence result

Product-form stationary distribution

$$\log p(s|\theta) = \log \rho(\theta)^\intercal x(s) - \log Z(\theta)$$

$$\downarrow$$

$$\nabla \log p(s|\theta) = \mathrm{D} \log \rho(\theta)^\intercal (x(s) - \mathbb{E}[x(S)])$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - (Ongoing) Run extensive numerical results on larger and more challenging examples.
  - Find better estimators for covariance and expectation, such as robust estimators.
  - Apply to MDPs where the stationary distribution is known only *up to a multiplicative constant*.

June–December 2024 in Toulouse

- 5 international workshops
- Atelier en Évaluation des Performances
- Invited researchers: Vivek Borkar, Itai Gurvich, Sean Meyn, and Adam Wierman

Call for abstract for the RL workshop until February 29!

More information on the webpage

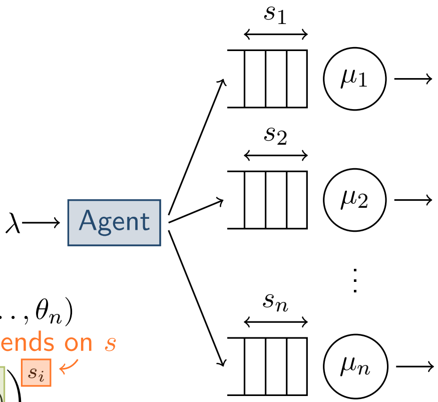
`https://indico.math.cnrs.fr/category/683/`

Example: Static load balancing

- Jobs arrive as a Poisson process with rate λ
- M/M/1 queues with service rates μ_1, μ_2, ..., μ_n
- Maximum c jobs in the system
- State vector $s = (s_1, s_2, \ldots, s_n)$ of queue sizes
- Actions are to assign to some server i
- Admission reward 1 per job
- Policy $\pi(\text{server } i | \cdot, \theta) = \dfrac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}}$ with $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$
- Stationary distribution $p(s|\theta) \propto \prod_{i=1}^{n} \left(\dfrac{\lambda}{\mu} \pi(\text{server } i | \cdot, \theta) \right)^{s_i}$

Depends on s

Depends on θ



Example: Static load balancing

Simulation setup

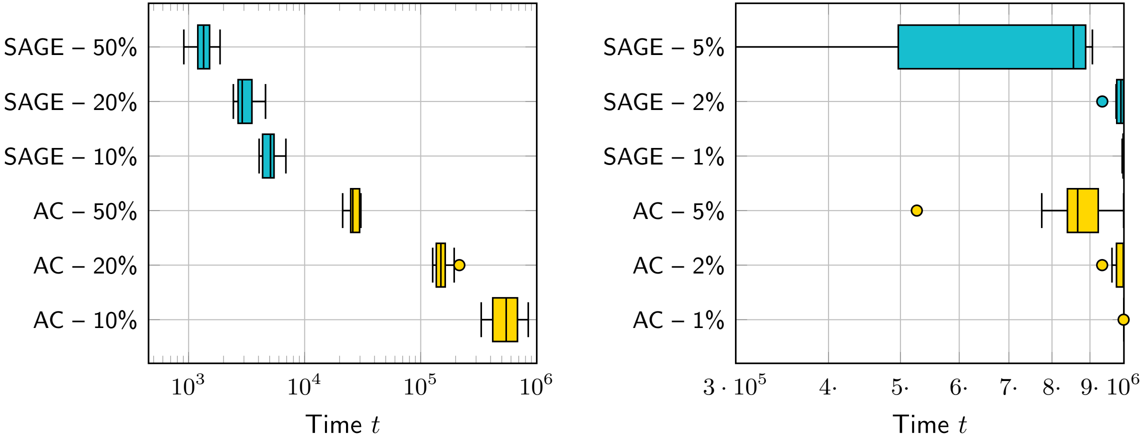
- Number $n = 4$ of servers
- Arrival rate $\lambda = 0.7$
- Service rates $\mu_1 = 0.4$, $\mu_2 = 0.3$, $\mu_3 = 0.2$, and $\mu_4 = 0.1$
- Total capacity $c = 10$
- 10^6 simulation steps

Algorithms

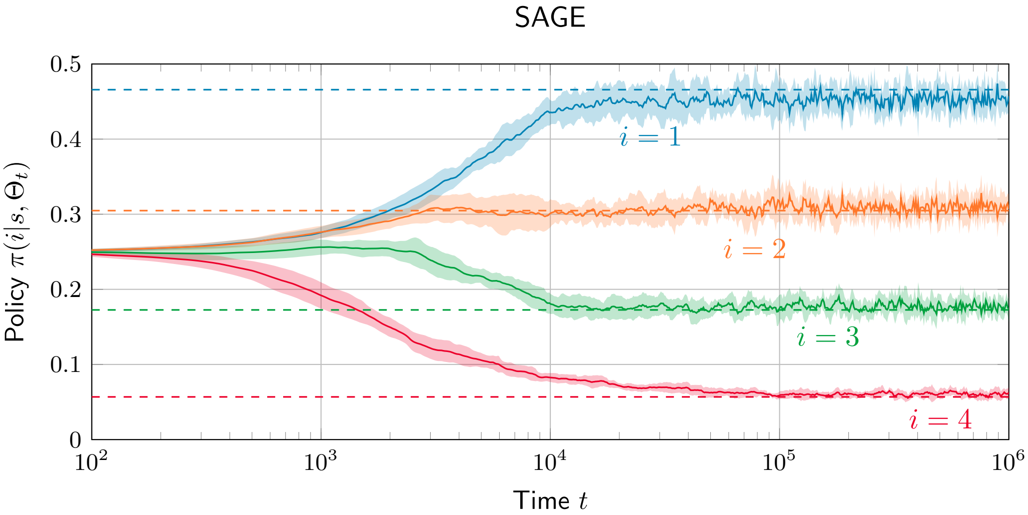
- SAGE-based policy-gradient
- Actor-critic without eligibility traces (Sutton and Barto, Section 13.6)
- Gradient update at every step with step size $\alpha = 10^{-3}$

Example: Static load balancing

Convergence times



Example: Static load balancing



Example: Static load balancing