Tailoring Policy Gradient to Product-Form Queueing Systems

Céline Comte and Matthieu Jonckheere
LAAS–CNRS and CNRS

Jaron Sanders and Albert Senen-Cerda
Eindhoven University of Technology

Work in Progress

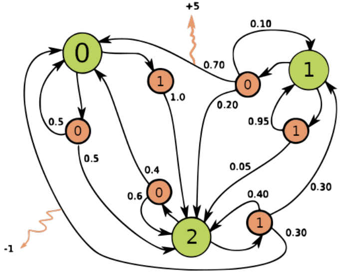
SOLACE Seminar – June 8, 2023

Reinforcement Learning

- A **agent** learns an optimal **policy** by interacting with an **environment** that sends **rewards**

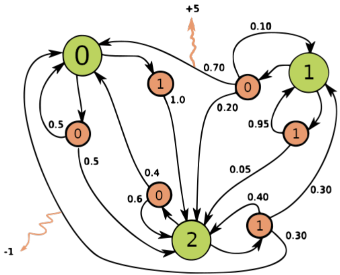
Reinforcement Learning

- A **agent** learns an optimal **policy** by interacting with an **environment** that sends **rewards**
- **Markov decision process** with a sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$



Source: Wikipedia (modified)

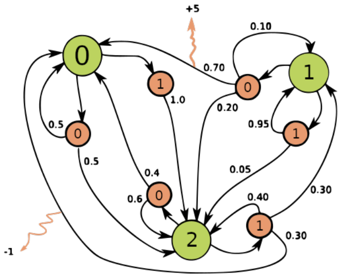
- A **agent** learns an optimal **policy** by interacting with an **environment** that sends **rewards**
- **Markov decision process** with a sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$
- Environment $P(s', r | s, a) = \mathbb{P}\left[\begin{matrix} S_{t+1}=s' \\ R_{t+1}=r \end{matrix} \middle| \begin{matrix} S_t=s \\ A_t=a \end{matrix}\right]$



Source: Wikipedia (modified)

Reinforcement Learning

- A **agent** learns an optimal **policy** by interacting with an **environment** that sends **rewards**
- **Markov decision process** with a sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$
- Environment $P(s', r|s, a) = \mathbb{P}\left[\begin{matrix} S_{t+1}=s' \\ R_{t+1}=r \end{matrix} \middle| \begin{matrix} S_t=s \\ A_t=a \end{matrix}\right]$
- Policy $\pi(a|s, \theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$

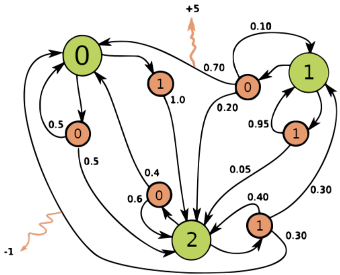


Source: Wikipedia (modified)

Reinforcement Learning

- A **agent** learns an optimal **policy** by interacting with an **environment** that sends **rewards**
- **Markov decision process** with a sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$
- Environment $P(s', r|s, a) = \mathbb{P}\left[\begin{smallmatrix} S_{t+1}=s' \\ R_{t+1}=r \end{smallmatrix} \,\middle|\, \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
- Policy $\pi(a|s, \theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$
- **Goal:** Find θ that maximizes the **average reward rate**

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

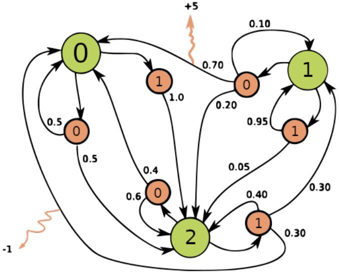


Source: Wikipedia (modified)

Reinforcement Learning

- A **agent** learns an optimal **policy** by interacting with an **environment** that sends **rewards**
- **Markov decision process** with a sequence $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$
- Environment $P(s', r|s, a) = \mathbb{P}\left[\begin{smallmatrix} S_{t+1}=s' \\ R_{t+1}=r \end{smallmatrix} \middle| \begin{smallmatrix} S_t=s \\ A_t=a \end{smallmatrix}\right]$
- Policy $\pi(a|s, \theta) = \mathbb{P}[A_t = a \,|\, S_t = s]$
- **Goal:** Find θ that maximizes the **average reward rate**

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

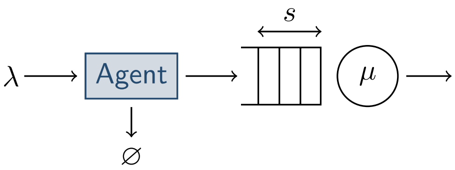


Source: Wikipedia (modified)

- $S \sim p(\cdot|\theta)$ stationary distribution of $(S_t, t = 0, 1, 2, \ldots)$
  $(S, A, R) \sim$ stationary distribution of $((S_t, A_t, R_{t+1}), t = 0, 1, 2, \ldots)$

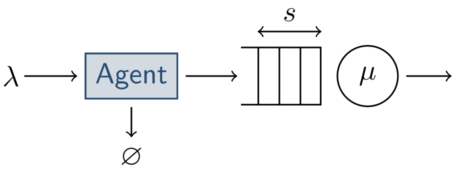
Example 1: M/M/1 Queue with Admission Control

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



Example 1: M/M/1 Queue with Admission Control

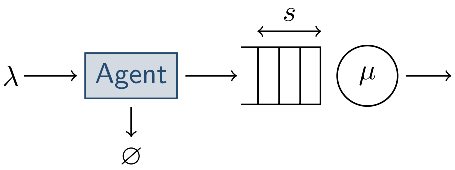
- Arrival rate $\lambda > 0$, service rate $\mu > \lambda$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Reward α per accepted job
- Holding cost η per job per time unit



- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\mathrm{accept}|s, \theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \frac{1}{\lambda}$

Example 1: M/M/1 Queue with Admission Control

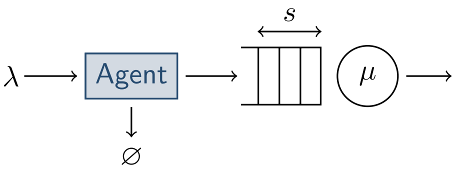
- Arrival rate $\lambda > 0$, service rate $\mu > \lambda$
- State: queue length $s \in \{0, 1, 2, \dots\}$
- Actions: accept or reject
- Reward α per accepted job
- Holding cost η per job per time unit



- Average reward rate $J(\theta) = \alpha \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{accept}|s, \theta) \right)}_{\text{Probabily of accepting a job}} - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \frac{1}{\lambda}$

Example 1: M/M/1 Queue with Admission Control

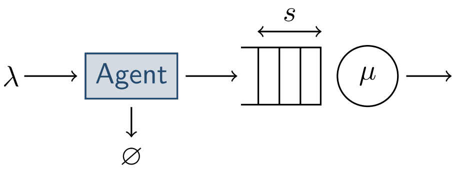
- Arrival rate $\lambda > 0$, service rate $\mu > \lambda$
- State: queue length $s \in \{0, 1, 2, \ldots\}$
- Actions: accept or reject
- Reward α per accepted job
- Holding cost η per job per time unit



- Average reward rate $J(\theta) = \alpha \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{accept}|s, \theta) \right)}_{\text{Probabily of accepting a job}} - \eta \times \underbrace{\left(\sum_{s=0}^{+\infty} p(s|\theta) s \right)}_{\text{Mean queue size}} \times \frac{1}{\lambda}$

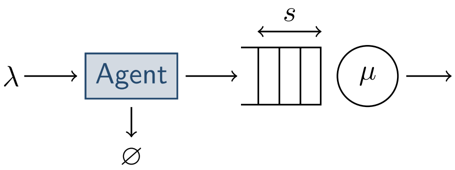
Example 1: M/M/1 Queue with Admission Control

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



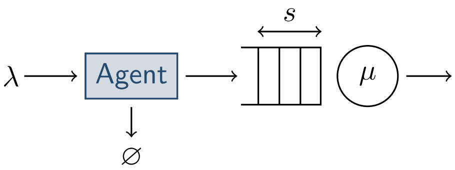
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{accept}|s, \theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \frac{1}{\lambda}$
- Policy $\pi(\text{accept}|s, \theta) = \frac{1}{1 + e^{-\theta_s}}$

Example 1: M/M/1 Queue with Admission Control



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

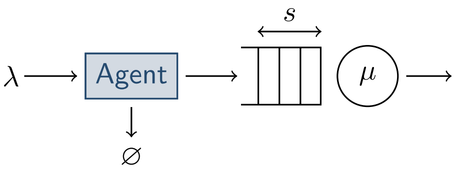
Example 1: M/M/1 Queue with Admission Control



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

Example 1: M/M/1 Queue with Admission Control

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

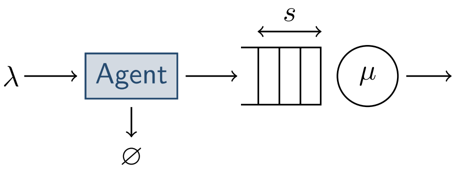


- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{accept}|s, \theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \frac{1}{\lambda}$
- Policy $\pi(\text{accept}|s, \theta) = \frac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter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

Depends on θ

Example 1: M/M/1 Queue with Admission Control

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



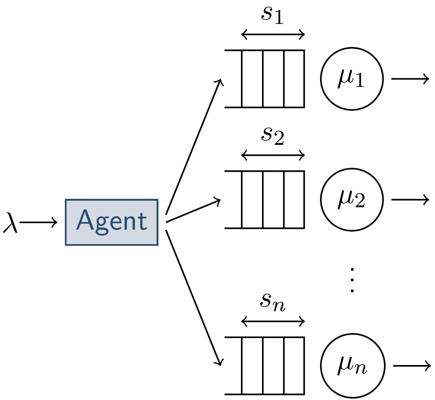
- Average reward rate $J(\theta) = \alpha \times \left(\sum_{s=0}^{+\infty} p(s|\theta) \pi(\text{accept}|s,\theta) \right) - \eta \times \left(\sum_{s=0}^{+\infty} p(s|\theta) s \right) \times \frac{1}{\lambda}$
- Policy $\pi(\text{accept}|s,\theta) = \frac{1}{1 + e^{-\theta_{\min(s,k)}}}$ with parameter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

Depends on s

Depends on θ

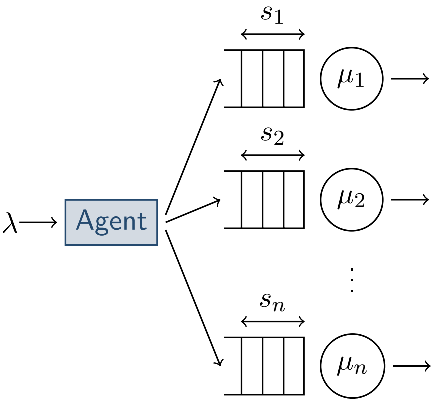
Example 2: Load Balancing

- Jobs arrive as a Poisson process with rate λ
- M/M/1 queues with service rates μ_1, μ_2, ..., μ_n
- Upper-bound ℓ on the number of jobs in the system
- State: vector $s = (s_1, s_2, \dots, s_n)$ of queue sizes
- Actions: assign to some server i
- Holding cost η per job per time unit



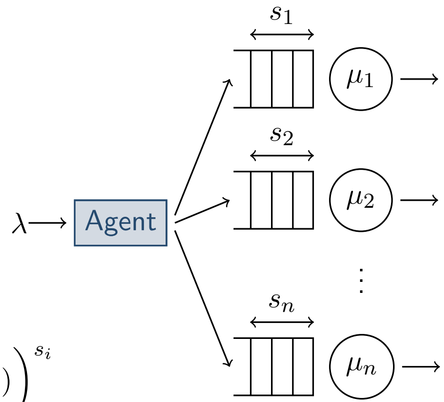
Example 2: Load Balancing

- Jobs arrive as a Poisson process with rate λ
- M/M/1 queues with service rates μ_1, μ_2, ..., μ_n
- Upper-bound ℓ on the number of jobs in the system
- State: vector $s = (s_1, s_2, \ldots, s_n)$ of queue sizes
- Actions: assign to some server i
- Holding cost η per job per time unit
- Policy $\pi(\text{server } i | \cdot, \theta) = \dfrac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}}$



Example 2: Load Balancing

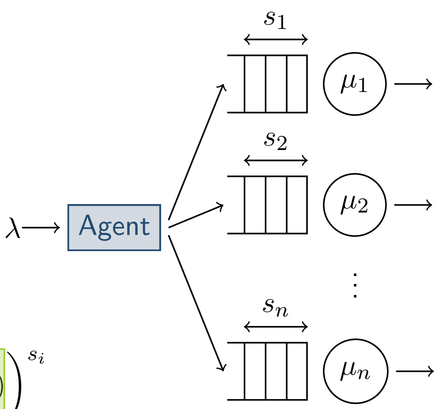
- Jobs arrive as a Poisson process with rate λ
- M/M/1 queues with service rates μ_1, μ_2, ..., μ_n
- Upper-bound ℓ on the number of jobs in the system
- State: vector $s = (s_1, s_2, \ldots, s_n)$ of queue sizes
- Actions: assign to some server i
- Holding cost η per job per time unit
- Policy $\pi(\text{server } i | \cdot, \theta) = \dfrac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}}$
- Stationary distribution $p(s|\theta) \propto \displaystyle\prod_{i=1}^{n} \left(\frac{\lambda}{\mu} \pi(\text{server } i | \cdot, \theta) \right)^{s_i}$



Example 2: Load Balancing

- Jobs arrive as a Poisson process with rate λ
- M/M/1 queues with service rates μ_1, μ_2, ..., μ_n
- Upper-bound ℓ on the number of jobs in the system
- State: vector $s = (s_1, s_2, \ldots, s_n)$ of queue sizes
- Actions: assign to some server i
- Holding cost η per job per time unit
- Policy $\pi(\text{server } i|\cdot, \theta) = \dfrac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=1}^{n} \left(\frac{\lambda}{\mu} \pi(\text{server } i|\cdot, \theta) \right)^{s_i}$

Depends on θ

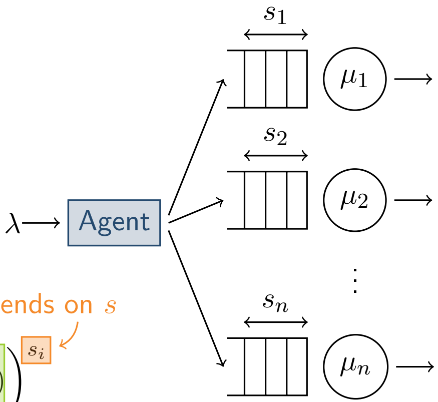


Example 2: Load Balancing

- Jobs arrive as a Poisson process with rate λ
- M/M/1 queues with service rates μ_1, μ_2, ..., μ_n
- Upper-bound ℓ on the number of jobs in the system
- State: vector $s = (s_1, s_2, \ldots, s_n)$ of queue sizes
- Actions: assign to some server i
- Holding cost η per job per time unit
- Policy $\pi(\text{server } i|\cdot, \theta) = \dfrac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}}$
- Stationary distribution $p(s|\theta) \propto \prod_{i=1}^{n} \left(\frac{\lambda}{\mu} \pi(\text{server } i|\cdot, \theta) \right)^{s_i}$

Depends on s

Depends on θ



Our Approach

- We consider MDPs and policy families $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t = 0, 1, 2, \ldots)$ has a **product-form stationary distribution** $p(s|\theta)$

Our Approach

- We consider MDPs and policy families $\pi(a|s, \theta)$ such that the Markov chain $(S_t, t = 0, 1, 2, \dots)$ has a **product-form stationary distribution** $p(s|\theta)$
- We exploit this product form to introduce a new **reinforcement learning algorithm**

Our Approach

- We consider MDPs and policy families $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t = 0, 1, 2, \ldots)$ has a **product-form stationary distribution** $p(s|\theta)$
- We exploit this product form to introduce a new **reinforcement learning algorithm**
- We show that this algorithm has nice **convergence properties**

Our Approach

- We consider MDPs and policy families $\pi(a|s,\theta)$ such that the Markov chain $(S_t, t = 0, 1, 2, \dots)$ has a **product-form stationary distribution** $p(s|\theta)$
- We exploit this product form to introduce a new **reinforcement learning algorithm**
- We show that this algorithm has nice **convergence properties**

- Main contributions:
  1. Product-form distributions as exponential families
  2. Score-aware gradient estimator (SAGE)
  3. SAGE-based policy-gradient algorithm
  4. Convergence result (work in progress)

① Product-Form Distributions as Exponential Families

- **Product-form** distribution

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

① Product-Form Distributions as Exponential Families

- **Product-form** distribution

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

Depends on θ

- **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

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

- **Feature function** $x = (x_1, x_2, \ldots, x_n)$

① Product-Form Distributions as Exponential Families

- **Product-form** distribution

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

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

① Product-Form Distributions as Exponential Families

- **Product-form** distribution

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

- **Feature function** $x = (x_1, x_2, \dots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$
- **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 ⟶ **Exponential family** of distributions

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

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \ldots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$
- **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 ⟶ **Exponential family** of distributions

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

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \ldots, x_n)$ ⟶ **Feature function** $x = (x_1, x_2, \ldots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$
- **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 ⟶ **Exponential family** of distributions

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

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \dots, x_n)$ ⟶ **Feature function** $x = (x_1, x_2, \dots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \dots, \rho_n)$ ⟶ **Log-load function** $\log \rho = (\log \rho_1, \dots, \log \rho_n)$
- **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 ⟶ **Exponential family** of distributions

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

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

- **Feature function** $x = (x_1, x_2, \ldots, x_n)$ ⟶ **Feature function** $x = (x_1, x_2, \ldots, x_n)$
- **Load function** $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$ ⟶ **Log-load function** $\log \rho = (\log \rho_1, \ldots, \log \rho_n)$
- **Partition function** Z ⟶ **Log-partition function** $\log Z$

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

$$\log Z(\theta) = \log \left(\sum_s \mathrm{e}^{\langle \log \rho(\theta), x(s) \rangle} \right)$$

② Score-aware gradient estimator (SAGE)

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

$$\text{“Likelihood”} = p(s|\theta) \quad \rightarrow \quad \text{“Score”} = \nabla_\theta \log p(s|\theta) = (\partial_{\theta_i} \log p(s|\theta), i = 1, 2, \ldots, n)$$

② Score-aware gradient estimator (SAGE)

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

$$\text{“Likelihood”} = p(s|\theta) \quad \rightarrow \quad \text{“Score”} = \nabla_\theta \log p(s|\theta) = (\partial_{\theta_i} \log p(s|\theta), i = 1, 2, \dots, n)$$

Theorem

Recalling that $(S, A, R) \sim$ *stationary distribution of* $((S_t, A_t, R_{t+1}), t = 0, 1, 2, \dots)$*, we have*

$$\partial_{\theta_i} \log p(s|\theta) = \langle \partial_{\theta_i} \log \rho(\theta), x(s) - \mathbb{E}[x(S)] \rangle,$$
$$\partial_{\theta_i} J(\theta) = \langle \partial_{\theta_i} \log \rho(\theta), \mathrm{Cov}[x(S), R] \rangle + \mathbb{E}[\partial_{\theta_i} \log \pi(A|S, \theta) R].$$

② Score-aware gradient estimator (SAGE)

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

$$\text{“Likelihood”} = p(s|\theta) \quad \to \quad \text{“Score”} = \nabla_\theta \log p(s|\theta) = (\partial_{\theta_i} \log p(s|\theta), i = 1, 2, \ldots, n)$$

Theorem

Recalling that $(S, A, R) \sim$ *stationary distribution of* $((S_t, A_t, R_{t+1}), t = 0, 1, 2, \ldots)$, *we have*

$$\begin{aligned} \partial_{\theta_i} \log p(s|\theta) &= \langle \partial_{\theta_i} \log \rho(\theta), x(s) - \mathbb{E}[x(S)] \rangle, \\ \partial_{\theta_i} J(\theta) &= \langle \partial_{\theta_i} \log \rho(\theta), \mathrm{Cov}[x(S), R] \rangle + \mathbb{E}[\partial_{\theta_i} \log \pi(A|S, \theta) R]. \end{aligned}$$

- **Main take-away:** This gives us an estimator for $\nabla_\theta J(\theta) = (\partial_{\theta_i} J(\theta), i = 1, \ldots, n)$.

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 S_{t+1}, R_{t+1}
5: Estimate $[\![\nabla_\theta J(\theta_t)]\!]$ 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 $\theta_{t+1} \leftarrow \theta_t + \alpha [\![\nabla_\theta J(\theta_t)]\!]$
7: **end for**

③ SAGE-Based Policy-Gradient Algorithm

- Typical **policy-gradient algorithm**:

1: Initialize S_0 and θ_0
2: **for** $t = 0, 1, 2, \dots$ **do**
3: Sample $A_t \sim \pi(\cdot|S_t, \theta_t)$
4: Take action A_t and observe S_{t+1}, R_{t+1}
5: Estimate $[\![\nabla_\theta J(\theta_t)]\!]$ 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 $\theta_{t+1} \leftarrow \theta_t + \alpha [\![\nabla_\theta J(\theta_t)]\!]$ Estimate
7: **end for**

③ SAGE-Based Policy-Gradient Algorithm

- 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 S_{t+1}, R_{t+1}
    Estimate $[\![\nabla_\theta J(\theta_t)]\!]$ 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 $\theta_{t+1} \leftarrow \theta_t + \alpha [\![\nabla_\theta J(\theta_t)]\!]$ Estimate
end for

③ SAGE-Based Policy-Gradient Algorithm

- Typical **policy-gradient algorithm**:

  Initialize S_0 and θ_0
  for $t = 0, 1, 2, \dots$ **do**
      Sample $A_t \sim \pi(\cdot|S_t, \theta_t)$
      Take action A_t and observe S_{t+1}, R_{t+1}
      Estimate $[\![\nabla_\theta J(\theta_t)]\!]$ 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?
      Update $\theta_{t+1} \leftarrow \theta_t + \alpha [\![\nabla_\theta J(\theta_t)]\!]$ Estimate
  end for

- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$[\![\partial_{\theta_i} J(\theta_t)]\!] \leftarrow ([\![\mathbb{E}[R]]\!] - [\![v]\!](S_t))\partial_{\theta_i} \log \pi(A_t|S_t, \theta_t).$$

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 S_{t+1}, R_{t+1}
5: Estimate $[\![\nabla_\theta J(\theta_t)]\!]$ 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 $\theta_{t+1} \leftarrow \theta_t + \alpha [\![\nabla_\theta J(\theta_t)]\!]$ Estimate
7: **end for**

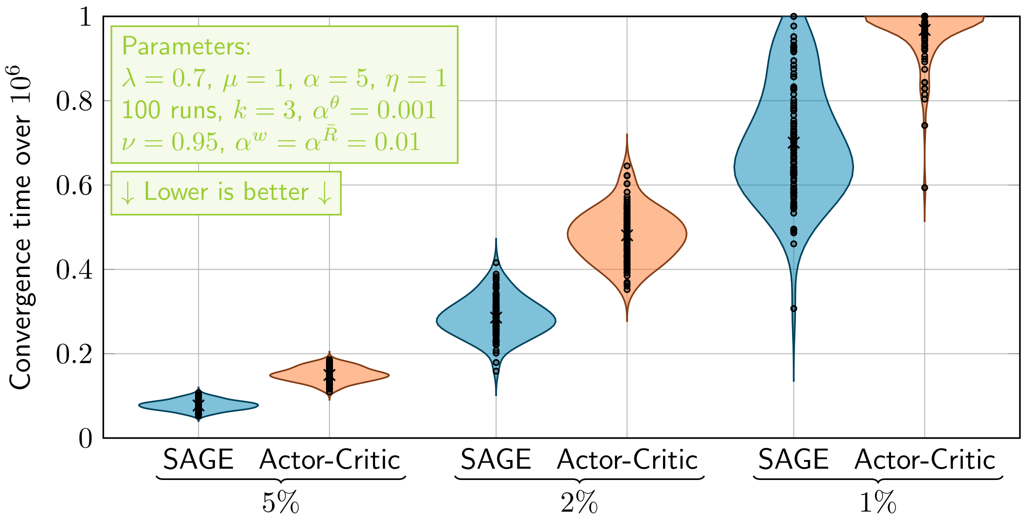
- **Actor-critic** applies the policy-gradient theorem (Sutton and Barto, 2018):

$$[\![\partial_{\theta_i} J(\theta_t)]\!] \leftarrow ([\![\mathbb{E}[R]]\!] - [\![v]\!](S_t))\partial_{\theta_i} \log \pi(A_t|S_t, \theta_t).$$

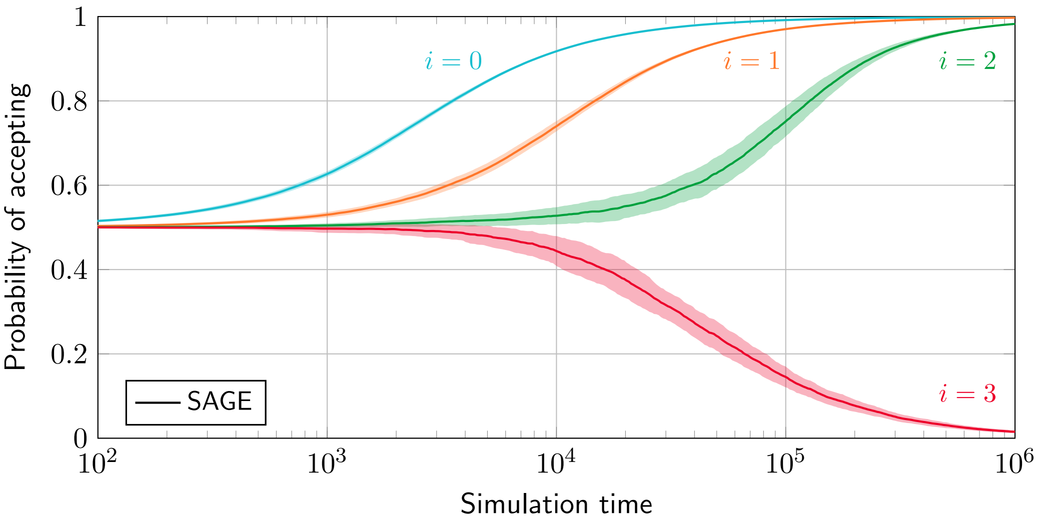
- We instead estimate $[\![\nabla_\theta J(\theta_t)]\!]$ with a **score-aware gradient estimator (SAGE)**:

$$[\![\partial_{\theta_i} J(\theta_t)]\!] \leftarrow \langle \partial_{\theta_i} \log \rho(\theta_t), [\![\mathrm{Cov}[x(S), R]]\!] \rangle + [\![\mathbb{E}[\partial_{\theta_i} \log \pi(A|S, \theta) R]]\!].$$

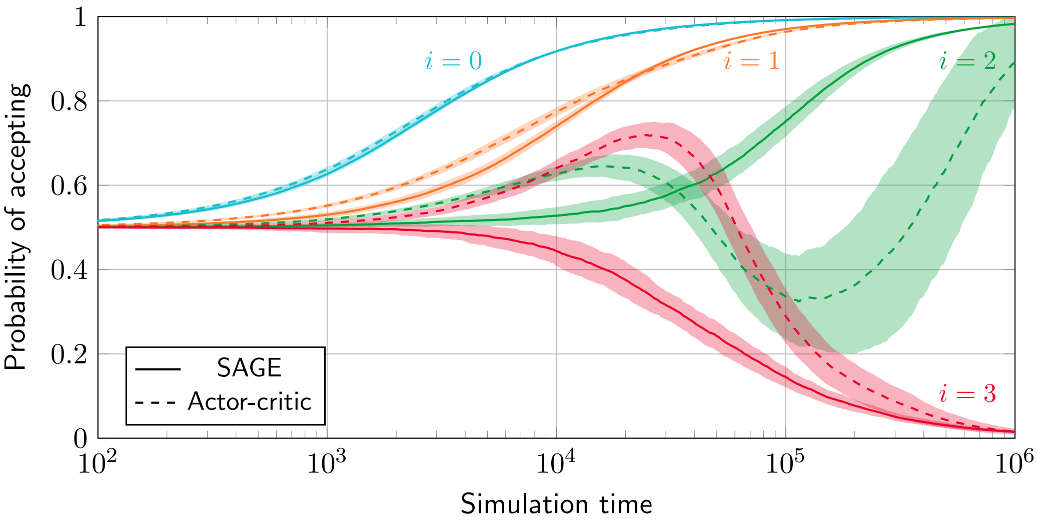
Example 1: M/M/1 Queue with Admission Control



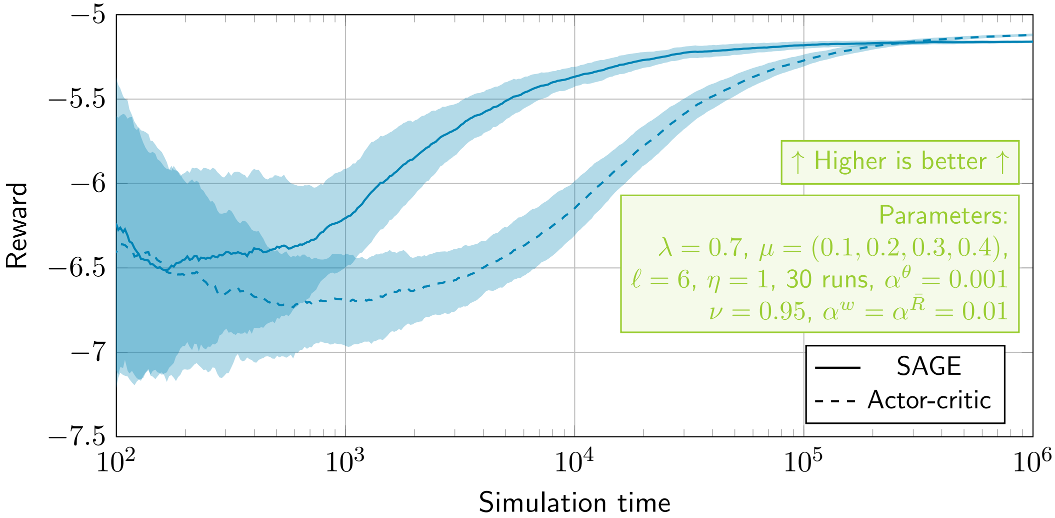
Example 1: M/M/1 Queue with Admission Control



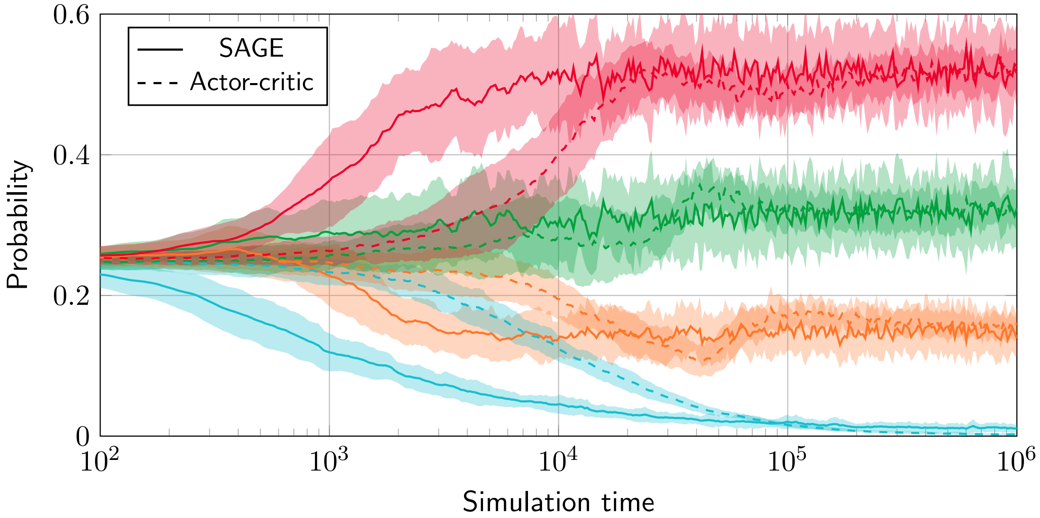
Example 1: M/M/1 Queue with Admission Control



Example 2: Load Balancing



Example 2: Load Balancing



Conclusion

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

Product-form stationary distribution

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

$$\downarrow$$

$$\partial_{\theta_i} \log p(s|\theta) = \langle \partial_{\theta_i} \log \rho(\theta), x(s) - \mathbb{E}[x(S)] \rangle$$

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. Convergence result (work in progress)

Product-form stationary distribution

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

$$\downarrow$$

$$\partial_{\theta_i} \log p(s|\theta) = \langle \partial_{\theta_i} \log \rho(\theta), x(s) - \mathbb{E}[x(S)] \rangle$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - Run extensive numerical results on 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. Convergence result (work in progress)

Product-form stationary distribution

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

$$\downarrow$$

$$\partial_{\theta_i} \log p(s|\theta) = \langle \partial_{\theta_i} \log \rho(\theta), x(s) - \mathbb{E}[x(S)] \rangle$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - Run extensive numerical results on more challenging examples.
  - Better estimators for covariance and expectation: robust covariance, etc.

Conclusion

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

Product-form stationary distribution

$$\log p(s|\theta) = \langle \log \rho(\theta), x(s) \rangle - \log Z(\theta)$$

$$\downarrow$$

$$\partial_{\theta_i} \log p(s|\theta) = \langle \partial_{\theta_i} \log \rho(\theta), x(s) - \mathbb{E}[x(S)] \rangle$$

Score-aware gradient estimator (SAGE)

- **Future research directions**
  - Run extensive numerical results on more challenging examples.
  - Better estimators for covariance and expectation: robust covariance, etc.
  - Applications to (queueing) systems where the stationary distribution is known only *up to a multiplicative constant*.