Stochastic approximation of symmetric Nash equilibria in queueing games

Ran I. Snitkovsky[1,2]

co-authored with

Liron Ravner [3]

[1]SRIBD, CUHK Shenzhen
[2]Columbia Business School
[3]Department of Statistics, University of Haifa

2021

Symmetric unobservable queueing games

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action $\Rightarrow$ Undergoes processing

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action $\Rightarrow$ Undergoes processing $\Rightarrow$ Receives utility

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action $\Rightarrow$ Undergoes processing $\Rightarrow$ Receives utility $\Rightarrow$ Leaves

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action $\Rightarrow$ Undergoes processing $\Rightarrow$ Receives utility $\Rightarrow$ Leaves

Customers are homogeneous.

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives	$\Rightarrow$	Takes action	$\Rightarrow$	Undergoes processing	$\Rightarrow$	Receives utility	$\Rightarrow$	Leaves

Customers are homogeneous.

The set of possible actions is finite.

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action $\Rightarrow$ Undergoes processing $\Rightarrow$ Receives utility $\Rightarrow$ Leaves

Customers are homogeneous.

The set of possible actions is finite.

A **strategy**

Symmetric unobservable queueing games

An infinite cohort of short-lived strategic customers

Customer arrives $\Rightarrow$ Takes action $\Rightarrow$ Undergoes processing $\Rightarrow$ Receives utility $\Rightarrow$ Leaves

Customers are homogeneous.

The set of possible actions is finite.

A **strategy** is a distribution over actions.

Symmetric unobservable queueing games

Customer utility depends on:

Symmetric unobservable queueing games

Customer utility depends on:

Their action

Symmetric unobservable queueing games

Customer utility depends on:

- Their action
- The population strategy

Symmetric unobservable queueing games

Customer utility depends on:

- Their action
- The population strategy
- (Steady-)state realization

Symmetric unobservable queueing games

Customer utility depends on:

- Their action
- The population strategy
- (Steady-)state realization
- Other random outcomes

Symmetric unobservable queueing games

Customer utility depends on:

- Their action
- The population strategy
- (Steady-)state realization
- Other random outcomes

A **solution**

Symmetric unobservable queueing games

Customer utility depends on:

- Their action
- The population strategy
- (Steady-)state realization
- Other random outcomes

A **solution** is a *Symmetric Nash Equilibrium* (SNE) strategy,

Symmetric unobservable queueing games

Customer utility depends on:

- Their action
- The population strategy
- (Steady-)state realization
- Other random outcomes

A **solution** is a *Symmetric Nash Equilibrium* (SNE) strategy, i.e., a strategy from which no customer has incentive to deviate.

Unobservable M/M/1 Model

Unobservable M/M/1 Model

An M/M/1 system

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions:

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1 - p)$,

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$\begin{array}{c}\text{util. from}\\\text{JOIN}\end{array} = R + C \times \begin{array}{c}\text{waiting}\\\text{time}\end{array}$$

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$\begin{array}{c} \text{util. from} \\ \text{JOIN} \end{array} = R + C \times \begin{array}{c} \text{waiting} \\ \text{time} \end{array}$$

$$\begin{array}{c} \text{util. from} \\ \text{BALK} \end{array} = 0$$

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$\begin{array}{c} \text{util. from} \\ \text{JOIN} \end{array} = R + C \times \begin{array}{c} \text{waiting} \\ \text{time} \end{array}$$

$$\begin{array}{c} \text{util. from} \\ \text{BALK} \end{array} = 0$$

Customers wish to maximize expected steady-state utility.

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$u_1(\mathbf{p}) = R + C \times \text{waiting time}$$

$$\text{util. from BALK} = 0$$

Customers wish to maximize expected steady-state utility.

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$u_1(\mathbf{p}) = R + C \times \mathrm{E}_{\mathbf{p}} \left(\begin{array}{c} \text{waiting} \\ \text{time} \end{array} \right)$$

$$\begin{array}{c} \text{util. from} \\ \text{BALK} \end{array} = 0$$

Customers wish to maximize expected steady-state utility.

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$u_1(\mathbf{p}) = R + C \times \mathrm{E}_{\mathbf{p}} \begin{pmatrix} \text{waiting} \\ \text{time} \end{pmatrix}$$

$$u_2(\mathbf{p}) = 0$$

Customers wish to maximize expected steady-state utility.

Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim \mathrm{Poisson}(\lambda)$

Possible actions: Join or Balk

A strategy $\mathbf{p} = (p, 1-p)$, with p = probability of Join

$$
\begin{aligned}
u_1(\mathbf{p}) &= R + C \times \mathrm{E}_{\mathbf{p}} \begin{pmatrix} \text{waiting} \\ \text{time} \end{pmatrix} \\
u_2(\mathbf{p}) &= 0
\end{aligned}
$$

Customers wish to maximize expected steady-state utility.

The vector of expected utilities: $\mathbf{u}(\mathbf{p}) = (u_1(\mathbf{p}), u_2(\mathbf{p}))$

Unobservable M/M/1 Model

We wish to identify a SNE strategy $\mathbf{p}^e = (p^e, 1 - p^e)$

Unobservable M/M/1 Model

We wish to identify a SNE strategy $\mathbf{p}^e = (p^e, 1 - p^e)$

$$\mathbf{p}^e \in \arg\max_{\mathbf{p} \in \boldsymbol{\Delta}} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

Unobservable M/M/1 Model

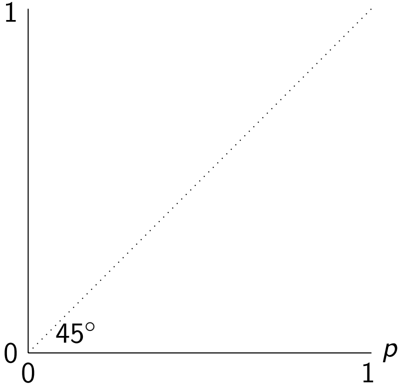
We wish to identify a SNE strategy $\mathbf{p}^e = (p^e, 1 - p^e)$

$$\mathbf{p}^e \in \arg\max_{\mathbf{p} \in \boldsymbol{\Delta}} \mathbf{p}' \mathbf{u}(\mathbf{p}^e) =: \mathcal{BR}(\mathbf{p}^e)$$

Unobservable M/M/1 Model

We wish to identify a SNE strategy $\mathbf{p}^e = (p^e, 1 - p^e)$

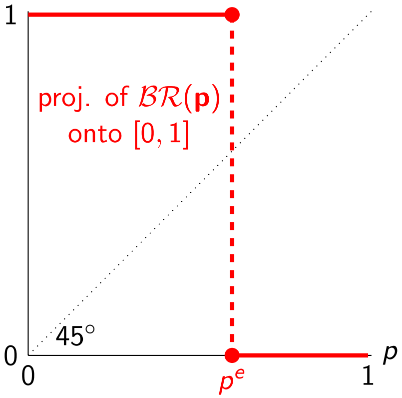
$$\mathbf{p}^e \in \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e) =: \mathcal{BR}(\mathbf{p}^e)$$



Unobservable M/M/1 Model

We wish to identify a SNE strategy $\mathbf{p}^e = (p^e, 1 - p^e)$

$$\mathbf{p}^e \in \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e) =: \mathcal{BR}(\mathbf{p}^e)$$



Unobservable M/M/1 Model

In this model $\mathbf{p}^e$ is available in closed form, solving:

Unobservable M/M/1 Model

In this model $\mathbf{p}^e$ is available in closed form, solving:

$$u_1(\mathbf{p}) = u_2(\mathbf{p})$$

Unobservable M/M/1 Model

In this model $\mathbf{p}^e$ is available in closed form, solving:

$$u_1(\mathbf{p}) = \underbrace{u_2(\mathbf{p})}_{=0}$$

Unobservable M/M/1 Model

In this model $\mathbf{p}^e$ is available in closed form, solving:

$$u_1(\mathbf{p}) = \underbrace{u_2(\mathbf{p})}_{=0}$$

...and similarly, for unobservable M/G/1 and G/M/1 models.

Unobservable M/M/1 Model

In this model $\mathbf{p}^e$ is available in closed form, solving:

$$u_1(\mathbf{p}) = \underbrace{u_2(\mathbf{p})}_{=0}$$

...and similarly, for unobservable M/G/1 and G/M/1 models.

What about a GI/G/1 queue?

Challenges and objective

Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available.

Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available.

$\mathcal{BR}$ is not (lower hemi-)continuous.

Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available.

$\mathcal{BR}$ is not (lower hemi-)continuous.

Lengthy simulations to verify SNE conditions for many $\mathbf{p}$'s are impracticable.

Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available.

$\mathcal{BR}$ is not (lower hemi-)continuous.

Lengthy simulations to verify SNE conditions for many $\mathbf{p}$'s are impracticable.

Our goal:

Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available.

$\mathcal{BR}$ is not (lower hemi-)continuous.

Lengthy simulations to verify SNE conditions for many $\mathbf{p}$'s are impracticable.

Our goal: Find a SNE strategy by running a *single* simulation of the system, with *dynamic updating* of the strategy.

Parallel GI/G/1 queues

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions:

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$,

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E} Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E} Y_2$$

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E} Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E} Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Actions:

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Actions: Join queue 1,

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Actions: Join queue 1, Join queue 2

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Actions: Join queue 1, Join queue 2 or Balk

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Actions: Join queue 1, Join queue 2 or Balk

Queue lengths are not observed upon arrival.

Parallel GI/G/1 queues

Two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means

$$\mathrm{E}Y_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = \mathrm{E}Y_2$$

Independent inter-arrival time distribution H with mean $\frac{1}{\lambda}$

Reward for obtaining service: $R > 0$

Cost per unit of waiting time: $C > 0$

Actions: Join queue 1, Join queue 2 or Balk

Queue lengths are not observed upon arrival.

The model is simple but completely intractable!

Parallel GI/G/1 queues

Denote the action set:

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{\text{Join 1}, \text{Join 2}, \text{Balk}\ \ \}$$

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

A strategy is a distribution $\mathbf{p} = (p_1, p_2, p_3)$ over $\mathcal{A}$.

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

A strategy is a distribution $\mathbf{p} = (p_1, p_2, p_3)$ over $\mathcal{A}$.

For queue $m \in \{1, 2\}$:

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

A strategy is a distribution $\mathbf{p} = (p_1, p_2, p_3)$ over $\mathcal{A}$.

For queue $m \in \{1, 2\}$:

If $\lambda p_m < \mu_m$,

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

A strategy is a distribution $\mathbf{p} = (p_1, p_2, p_3)$ over $\mathcal{A}$.

For queue $m \in \{1, 2\}$:

If $\lambda p_m < \mu_m$, then the stationary workload W_m exists.

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

A strategy is a distribution $\mathbf{p} = (p_1, p_2, p_3)$ over $\mathcal{A}$.

For queue $m \in \{1, 2\}$:

If $\lambda p_m < \mu_m$, then the stationary workload W_m exists.

If $\mathrm{E} Y_m^2 < \infty$,

Parallel GI/G/1 queues

Denote the action set:

$$\mathcal{A} = \{ \quad a_1 \quad , \quad a_2 \quad , \quad a_3 \quad \}$$

A strategy is a distribution $\mathbf{p} = (p_1, p_2, p_3)$ over $\mathcal{A}$.

For queue $m \in \{1, 2\}$:

> If $\lambda p_m < \mu_m$, then the stationary workload W_m exists.
>
> If $\mathrm{E} Y_m^2 < \infty$, then $w_m(p_m) := \mathrm{E}_{\mathbf{p}} W_m < \infty$.

Parallel GI/G/1 queues

The SNE condition:

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e)$$

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e) = \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e) = \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

where

$$\mathbf{u}(\mathbf{p}) =$$

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e) = \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

where

$$\mathbf{u}(\mathbf{p}) = \begin{pmatrix} R - C\cdot(w_1(p_1) + 1/\mu_1) \\ R - C\cdot(w_2(p_2) + 1/\mu_2) \\ 0 \end{pmatrix}$$

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e) = \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

where

$$\mathbf{u}(\mathbf{p}) = \begin{pmatrix} R - C\cdot(w_1(p_1) + 1/\mu_1) \\ R - C\cdot(w_2(p_2) + 1/\mu_2) \\ 0 \end{pmatrix}$$

It can be verified that $\mathbf{p}_e$ exists uniquely.

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e) = \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

where

$$\mathbf{u}(\mathbf{p}) = \begin{pmatrix} R - C \cdot (w_1(p_1) + 1/\mu_1) \\ R - C \cdot (w_2(p_2) + 1/\mu_2) \\ 0 \end{pmatrix}$$

It can be verified that $\mathbf{p}_e$ exists uniquely.

However, an expression for $w_m(p_m)$ is not available.

Parallel GI/G/1 queues

The SNE condition:

$$\mathbf{p}^e \in \mathcal{BR}(\mathbf{p}^e) = \arg\max_{\mathbf{p}\in\Delta} \mathbf{p}'\mathbf{u}(\mathbf{p}^e)$$

where

$$\mathbf{u}(\mathbf{p}) = \begin{pmatrix} R - C \cdot (w_1(p_1) + 1/\mu_1) \\ R - C \cdot (w_2(p_2) + 1/\mu_2) \\ 0 \end{pmatrix}$$

It can be verified that $\mathbf{p}_e$ exists uniquely.

However, an expression for $w_m(p_m)$ is not available.

How to compute $\mathbf{p}_e$?

SA Algorithm

SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm.

SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm.

Regeneration cycle:

SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm.

Regeneration cycle: the time between two arrival instants to empty system.

SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm.

Regeneration cycle: the time between two arrival instants to empty system.

Cycle length:

SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm.

Regeneration cycle: the time between two arrival instants to empty system.

Cycle length: the number of arrivals during a cycle (including balkings).

SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm.

Regeneration cycle: the time between two arrival instants to empty system.

Cycle length: the number of arrivals during a cycle (including balkings).

Our stability assumptions imply that the cycle length is finite (a.s.)

SA Algorithm

At iteration $n \geq 1$ of the algorithm:

SA Algorithm

At iteration $n \geq 1$ of the algorithm:

Given a strategy $\mathbf{p}^{(n)}$, generate 1 cycle.

SA Algorithm

At iteration $n \geq 1$ of the algorithm:

Given a strategy $\mathbf{p}^{(n)}$, generate 1 cycle.

Let L denote the cycle length.

SA Algorithm

At iteration $n \geq 1$ of the algorithm:

Given a strategy $\mathbf{p}^{(n)}$, generate 1 cycle.

Let L denote the cycle length.

Record the vector total expected utilities:

SA Algorithm

At iteration $n \geq 1$ of the algorithm:

Given a strategy $\mathbf{p}^{(n)}$, generate 1 cycle.

Let L denote the cycle length.

Record the vector total expected utilities:

$$\mathbf{G}^{(n)} = \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \sum_{j=1}^{L} \begin{pmatrix} R - C \cdot (X_j^{[1]} + 1/\mu_1) \\ R - C \cdot (X_j^{[2]} + 1/\mu_2) \\ 0 \end{pmatrix},$$

SA Algorithm

At iteration $n \geq 1$ of the algorithm:

Given a strategy $\mathbf{p}^{(n)}$, generate 1 cycle.

Let L denote the cycle length.

Record the vector total expected utilities:

$$\mathbf{G}^{(n)} = \begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} = \sum_{j=1}^{L} \begin{pmatrix} R - C \cdot (X_j^{[1]} + 1/\mu_1) \\ R - C \cdot (X_j^{[2]} + 1/\mu_2) \\ 0 \end{pmatrix},$$

where $X_j^{[m]}$ is the workload in queue $m = 1, 2$ at the j'th arrival.

SA Algorithm

Start with an arbitrary strategy $\mathbf{p}^{(0)}$,

SA Algorithm

Start with an arbitrary strategy $\mathbf{p}^{(0)}$, and initial step size $\gamma_0 > 0$.

SA Algorithm

Start with an arbitrary strategy $\mathbf{p}^{(0)}$, and initial step size $\gamma_0 > 0$.

Update the strategy as follows:

SA Algorithm

Start with an arbitrary strategy $\mathbf{p}^{(0)}$, and initial step size $\gamma_0 > 0$.

Update the strategy as follows:

$$\mathbf{p}^{(n+1)} = \mathbf{p}^{(n)} + \frac{\gamma_0}{n+1}\mathbf{G}^{(n)}.$$

SA Algorithm

Start with an arbitrary strategy $\mathbf{p}^{(0)}$, and initial step size $\gamma_0 > 0$.

Update the strategy as follows:

$$\mathbf{p}^{(n+1)} = \mathbf{p}^{(n)} + \frac{\gamma_0}{n+1}\mathbf{G}^{(n)}.$$

projecting onto Δ when necessary.

SA Algorithm

Start with an arbitrary strategy $\mathbf{p}^{(0)}$, and initial step size $\gamma_0 > 0$.

Update the strategy as follows:

$$\mathbf{p}^{(n+1)} = \mathbf{p}^{(n)} + \frac{\gamma_0}{n+1}\mathbf{G}^{(n)}.$$

projecting onto Δ when necessary.

It can be shown that $\mathbf{p}^{(n)} \to_{\mathrm{as}} \mathbf{p}^{e}$.

Simulation Results

Simulation Results

$F_1 \sim \mathrm{Beta}(10, 10) + 0.5,$

Simulation Results

$F_1 \sim \mathrm{Beta}(10, 10) + 0.5,\ F_2 \sim \mathrm{Bernoulli}(.1) \cdot 10,$

Simulation Results

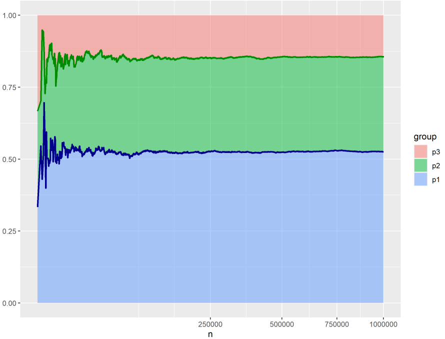
$F_1 \sim \mathrm{Beta}(10, 10) + 0.5$, $F_2 \sim \mathrm{Bernoulli}(.1) \cdot 10$, and $H \sim \mathrm{Gamma}(.1, 11)$

Simulation Results

$F_1 \sim \mathrm{Beta}(10, 10) + 0.5$, $F_2 \sim \mathrm{Bernoulli}(.1) \cdot 10$, and $H \sim \mathrm{Gamma}(.1, 11)$ with $R = 5$, $C = 1$

Simulation Results

$F_1 \sim \mathrm{Beta}(10, 10) + 0.5$, $F_2 \sim \mathrm{Bernoulli}(.1) \cdot 10$, and $H \sim \mathrm{Gamma}(.1, 11)$ with $R = 5$, $C = 1$



Coordinates of $\mathbf{p}^{(n)} = (p_1^{(n)}, p_2^{(n)}, p_3^{(n)})$ are plotted vs. n (square-root-scaled).

Simulation Results

$F_1 \sim \mathrm{Beta}(10, 10) + 0.5$, $F_2 \sim \mathrm{Bernoulli}(.1) \cdot 10$, and $H \sim \mathrm{Gamma}(.1, 11)$ with $R = 5$, $C = 1$

Coordinates of $\mathbf{p}^{(n)} = (p_1^{(n)}, p_2^{(n)}, p_3^{(n)})$ are plotted vs. n (square-root-scaled).

ε-equilibrium condition satisfied for $\varepsilon < 0.02$ with $> .99$ certainty

The Robbins-Monro algorithm

The Robbins-Monro algorithm

Goal:

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution:

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot g(\theta^{(n)})$$

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot g(\theta^{(n)})$$

E.g., in Gradient Descent

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot g(\theta^{(n)})$$

E.g., in Gradient Descent, $g(\theta) = f'(\theta)$

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot g(\theta^{(n)})$$

E.g., in Fixed-point Iteration,

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot g(\theta^{(n)})$$

E.g., in Fixed-point Iteration, $g(\theta) = f(\theta) - \theta$

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot g(\theta^{(n)})$$

E.g., in Fixed-point Iteration, $g(\theta) = f(\theta) - \theta$

The SA version (Robbins-Monro) mimics the deterministic one by plugging in an estimator instead of $g(\theta^{(n)})$

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot G^{(n)}$$

E.g., in Fixed-point Iteration, $g(\theta) = f(\theta) - \theta$

The SA version (Robbins-Monro) mimics the deterministic one by plugging in an estimator instead of $g(\theta^{(n)})$

The Robbins-Monro algorithm

Goal: Find the root of a continuous function $g : \mathbb{R} \to \mathbb{R}$.

Iterative solution: Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:

$$\theta^{(n+1)} = \theta^{(n)} + \gamma_n \cdot G^{(n)}$$

E.g., in Fixed-point Iteration, $g(\theta) = f(\theta) - \theta$

The SA version (Robbins-Monro) mimics the deterministic one by plugging in an estimator instead of $g(\theta^{(n)})$

Under mild regularity (unbiasedness & appropriate step sizes) the SA version converges a.s. to a root

Relevant literature

Relevant literature

Literature about unobservable queueing games is extensive.

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples:

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples: *Fu ('90, '94), L'Ecuyer and Glynn ('94a, '94b), Andradóttir ('96), Chen et al. (2020).*

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples: *Fu ('90, '94), L'Ecuyer and Glynn ('94a, '94b), Andradóttir ('96), Chen et al. (2020).*
Gap: Estimating gradients of performance measures is conceptually from finding SNE in a queueing game.

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples: *Fu ('90, '94), L'Ecuyer and Glynn ('94a, '94b), Andradóttir ('96), Chen et al. (2020).*
Gap: Estimating gradients of performance measures is conceptually from finding SNE in a queueing game.

An SA-like method was applied to justify equilibrium formation in a special Markoviran single-queue PS model:

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples: *Fu ('90, '94), L'Ecuyer and Glynn ('94a, '94b), Andradóttir ('96), Chen et al. (2020).*
Gap: Estimating gradients of performance measures is conceptually from finding SNE in a queueing game.

An SA-like method was applied to justify equilibrium formation in a special Markoviran single-queue PS model: **Examples:** *Altman and Shimkin ('98), Ben-Shahar et.al (2000), Buche and Kushner (2000).*

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples: *Fu ('90, '94), L'Ecuyer and Glynn ('94a, '94b), Andradóttir ('96), Chen et al. (2020).*
Gap: Estimating gradients of performance measures is conceptually from finding SNE in a queueing game.

An SA-like method was applied to justify equilibrium formation in a special Markoviran single-queue PS model: **Examples:** *Altman and Shimkin ('98), Ben-Shahar et.al (2000), Buche and Kushner (2000).*

Recent work applied reinforcement learning to find optimal policies.

Relevant literature

Literature about unobservable queueing games is extensive.
Overviews: *Hassin and Haviv (2006), Hassin (2016)*.
Gap: The focus is on stylized tractable systems.

SA methods are studied in the context of optimization in queues (e.g., optimizing capacity / pricing).
Examples: *Fu ('90, '94), L'Ecuyer and Glynn ('94a, '94b), Andradóttir ('96), Chen et al. (2020).*
Gap: Estimating gradients of performance measures is conceptually from finding SNE in a queueing game.

An SA-like method was applied to justify equilibrium formation in a special Markoviran single-queue PS model: **Examples:** *Altman and Shimkin ('98), Ben-Shahar et.al (2000), Buche and Kushner (2000).*

Recent work applied reinforcement learning to find optimal policies. **Examples:** *Dai and Gluzman (2020), Liu et al. (2019).*

General unobservable queueing game

General unobservable queueing game

Renewal arrival process (iid inter-arrivals).

General unobservable queueing game

Renewal arrival process (iid inter-arrivals).

Customers choose one of k actions:

General unobservable queueing game

Renewal arrival process (iid inter-arrivals).

Customers choose one of k actions: $\mathcal{A} = \{a_1, \ldots, a_k\}$.

General unobservable queueing game

Renewal arrival process (iid inter-arrivals).

Customers choose one of k actions: $\mathcal{A} = \{a_1, \ldots, a_k\}$.

The space of strategies is the $(k-1)$-dimensional simplex:

General unobservable queueing game

Renewal arrival process (iid inter-arrivals).

Customers choose one of k actions: $\mathcal{A} = \{a_1, \ldots, a_k\}$.

The space of strategies is the $(k-1)$-dimensional simplex:

$$\Delta = \left\{ \mathbf{p} : \ \forall i = 1, \ldots, k, \ p_i \geq 0, \sum_{i=1}^{k} p_i = 1 \right\}.$$

General unobservable queueing game

Renewal arrival process (iid inter-arrivals).

Customers choose one of k actions: $\mathcal{A} = \{a_1, \dots, a_k\}$.

The space of strategies is the $(k-1)$-dimensional simplex:

$$\Delta = \left\{ \mathbf{p} : \ \forall i = 1, \dots, k, \ p_i \geq 0, \sum_{i=1}^{k} p_i = 1 \right\}.$$

When all are playing strategy $\mathbf{p} \in \Delta$, denote the state at n'th arrival by $X_n(\mathbf{p}) \in \mathbb{R}$.

Regenerative structure

Regenerative structure

Assume the system starts empty:

Regenerative structure

Assume the system starts empty: $X_0(\mathbf{p}) = 0$.

Regenerative structure

Assume the system starts empty: $X_0(\mathbf{p}) = 0$.

$L(\mathbf{p}) = \inf\{n \geq 1 : \; X_n(\mathbf{p}) = 0\}$ is the cycle length.

Regenerative structure

Assume the system starts empty: $X_0(\mathbf{p}) = 0$.

$L(\mathbf{p}) = \inf\{n \geq 1 : \; X_n(\mathbf{p}) = 0\}$ is the cycle length.

Let $\ell^k(\mathbf{p}) = \mathrm{E}_{\mathbf{p}} L^k(\mathbf{p})$.

Regenerative structure

Assume the system starts empty: $X_0(\mathbf{p}) = 0$.

$L(\mathbf{p}) = \inf\{n \geq 1 : \ X_n(\mathbf{p}) = 0\}$ is the cycle length.

Let $\ell^k(\mathbf{p}) = \mathrm{E}_{\mathbf{p}} L^k(\mathbf{p})$.

Assume $\ell(\mathbf{p}) < \infty$ for any $\mathbf{p} \in \Delta$

Regenerative structure

Assume the system starts empty: $X_0(\mathbf{p}) = 0$.

$L(\mathbf{p}) = \inf\{n \geq 1 : \; X_n(\mathbf{p}) = 0\}$ is the cycle length.

Let $\ell^k(\mathbf{p}) = \mathrm{E}_{\mathbf{p}} L^k(\mathbf{p})$.

Assume $\ell(\mathbf{p}) < \infty$ for any $\mathbf{p} \in \Delta \Rightarrow$ there exists a stationary distribution, $X_n(\mathbf{p}) \; \rightarrow_{\mathrm{d}} \; X(\mathbf{p})$.

Regenerative structure

Assume the system starts empty: $X_0(\mathbf{p}) = 0$.

$L(\mathbf{p}) = \inf\{n \geq 1 : \; X_n(\mathbf{p}) = 0\}$ is the cycle length.

Let $\ell^k(\mathbf{p}) = \mathrm{E}_{\mathbf{p}} L^k(\mathbf{p})$.

Assume $\ell(\mathbf{p}) < \infty$ for any $\mathbf{p} \in \Delta \Rightarrow$ there exists a stationary distribution, $X_n(\mathbf{p}) \;\to_{\mathrm{d}}\; X(\mathbf{p})$.

Thus, the system is regenerative at 0 and is stable for any strategy.

Utility structure

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$,

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

y is some realized random outcome (e.g., service time),

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

- x is the (realized) state upon arrival,
- y is some realized random outcome (e.g., service time),
- $\mathbf{p}$ is the population strategy.

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

y is some realized random outcome (e.g., service time),

$\mathbf{p}$ is the population strategy.

Let $\mathbf{v}(x, y, \mathbf{p}) = (v_i(x, y, \mathbf{p}))_{i=1,\ldots,k}$.

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

y is some realized random outcome (e.g., service time),

$\mathbf{p}$ is the population strategy.

Let $\mathbf{v}(x, y, \mathbf{p}) = (v_i(x, y, \mathbf{p}))_{i=1,\ldots,k}$.

For example, in the unobservable M/M/1:

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

y is some realized random outcome (e.g., service time),

$\mathbf{p}$ is the population strategy.

Let $\mathbf{v}(x, y, \mathbf{p}) = (v_i(x, y, \mathbf{p}))_{i=1,\dots,k}$.

For example, in the unobservable M/M/1:

$$\mathbf{v}(x, y, \mathbf{p}) = \begin{pmatrix} v_1(x, y, \mathbf{p}) \\ v_2(x, y, \mathbf{p}) \end{pmatrix} = \begin{pmatrix} R - C(x + y) \\ 0 \end{pmatrix}$$

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

- x is the (realized) state upon arrival,
- y is some realized random outcome (e.g., service time),
- $\mathbf{p}$ is the population strategy.

Let $\mathbf{v}(x, y, \mathbf{p}) = (v_i(x, y, \mathbf{p}))_{i=1,\dots,k}$.

The mean stationary utility vector is

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

y is some realized random outcome (e.g., service time),

$\mathbf{p}$ is the population strategy.

Let $\mathbf{v}(x, y, \mathbf{p}) = (v_i(x, y, \mathbf{p}))_{i=1,\ldots,k}$.

The mean stationary utility vector is

$$\mathbf{u}(\mathbf{p}) = \mathrm{E}_{\mathbf{p}}\Big[\mathbf{v}\big(X(\mathbf{p}), Y, \mathbf{p}\big)\Big].$$

Utility structure

The utility of a customer choosing action a_i is $v_i(x, y, \mathbf{p})$, where:

x is the (realized) state upon arrival,

y is some realized random outcome (e.g., service time),

$\mathbf{p}$ is the population strategy.

Let $\mathbf{v}(x, y, \mathbf{p}) = (v_i(x, y, \mathbf{p}))_{i=1,\ldots,k}$.

The mean stationary utility vector is

$$\mathbf{u}(\mathbf{p}) = \mathrm{E}_{\mathbf{p}}\Big[\mathbf{v}\big(X(\mathbf{p}), Y, \mathbf{p}\big)\Big].$$

Each coordinate i of $\mathbf{u}(\mathbf{p})$ corresponds to the mean stationary utility from action a_i.

Symmetric Nash equilibrium

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

$$\mathbf{p} \in \arg\max_{\mathbf{q}\in\Delta} \mathbf{u}(\mathbf{p})'\mathbf{q}$$

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

$$\mathbf{p} \in \underbrace{\arg\max_{\mathbf{q} \in \Delta} \mathbf{u}(\mathbf{p})'\mathbf{q}}_{=: \mathcal{BR}(\mathbf{p})}$$

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

$$\mathbf{p} \in \underbrace{\arg\max_{\mathbf{q} \in \Delta} \mathbf{u}(\mathbf{p})'\mathbf{q}}_{=:\mathcal{BR}(\mathbf{p})}$$

Problem:

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

$$\mathbf{p} \in \underbrace{\arg\max_{\mathbf{q} \in \Delta} \mathbf{u}(\mathbf{p})'\mathbf{q}}_{=:\mathcal{BR}(\mathbf{p})}$$

Problem: $\mathcal{BR}(\mathbf{p})$ is a set-valued mapping, and not lower-hemicontinuous.

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

$$\mathbf{p} \in \underbrace{\arg\max_{\mathbf{q}\in\Delta} \mathbf{u}(\mathbf{p})'\mathbf{q}}_{=:\mathcal{BR}(\mathbf{p})}$$

Problem: $\mathcal{BR}(\mathbf{p})$ is a set-valued mapping, and not lower-hemicontinuous.

Solution:

Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy $\mathbf{p} \in \Delta$ such that

$$\mathbf{p} \in \underbrace{\arg\max_{\mathbf{q} \in \Delta} \mathbf{u}(\mathbf{p})'\mathbf{q}}_{=:\mathcal{BR}(\mathbf{p})}$$

Problem: $\mathcal{BR}(\mathbf{p})$ is a set-valued mapping, and not lower-hemicontinuous.

Solution: We use a surrogate best response function.

Best response surrogate

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$,

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

$$\phi(\mathbf{q}; \mathbf{p}) = \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2.$$

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

$$\phi(\mathbf{q}; \mathbf{p}) = \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2.$$

This yields a surrogate best response function:

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

$$\phi(\mathbf{q}; \mathbf{p}) = \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2.$$

This yields a surrogate best response function:

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q} \in \Delta} \phi(\mathbf{q}; \mathbf{p})$$

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

$$\phi(\mathbf{q}; \mathbf{p}) = \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2.$$

This yields a surrogate best response function:

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \phi(\mathbf{q}; \mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2 \right\}.$$

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

$$\phi(\mathbf{q}; \mathbf{p}) = \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2.$$

This yields a surrogate best response function:

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \phi(\mathbf{q}; \mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2 \right\}.$$

The function f fixes the discontinuities in $\mathcal{BR}$!

Best response surrogate

For a vector $\mathbf{q} \in \mathbb{R}^k$ and a strategy $\mathbf{p} \in \Delta$, define the function $\phi : \mathbb{R}^k \times \Delta \to \mathbb{R}$ as

$$\phi(\mathbf{q}; \mathbf{p}) = \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2.$$

This yields a surrogate best response function:

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \phi(\mathbf{q}; \mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p} - \mathbf{q}\|^2 \right\}.$$

The function f fixes the discontinuities in $\mathcal{BR}$!

The choice of $\ell(\mathbf{p})$ will be made clear later.

Alternative equilibrium condition

Alternative equilibrium condition

Lemma

Alternative equilibrium condition

Lemma

A strategy $\mathbf{p} \in \Delta$ *is a Symmetric Nash Equilibrium, i.e.,* $\mathbf{p} \in \mathcal{BR}(\mathbf{p})$,

Alternative equilibrium condition

Lemma

A strategy $\mathbf{p} \in \Delta$ *is a Symmetric Nash Equilibrium, i.e.,* $\mathbf{p} \in \mathcal{BR}(\mathbf{p})$*, if and only if it satisfies* $\mathbf{p} = \mathbf{f}(\mathbf{p})$*.*

Alternative equilibrium condition

Lemma

A strategy $\mathbf{p} \in \Delta$ *is a Symmetric Nash Equilibrium, i.e.,* $\mathbf{p} \in \mathcal{BR}(\mathbf{p})$, *if and only if it satisfies* $\mathbf{p} = \mathbf{f}(\mathbf{p})$.

Lemma

Alternative equilibrium condition

Lemma

A strategy $\mathbf{p} \in \Delta$ *is a Symmetric Nash Equilibrium, i.e.,* $\mathbf{p} \in \mathcal{BR}(\mathbf{p})$*, if and only if it satisfies* $\mathbf{p} = \mathbf{f}(\mathbf{p})$.

Lemma

Assume both $\mathbf{u}(\mathbf{p})$ *and* $\ell(\mathbf{p})$ *are continuous for all* $\mathbf{p} \in \Delta$.

Alternative equilibrium condition

Lemma

A strategy $\mathbf{p} \in \Delta$ *is a Symmetric Nash Equilibrium, i.e.,* $\mathbf{p} \in \mathcal{BR}(\mathbf{p})$*, if and only if it satisfies* $\mathbf{p} = \mathbf{f}(\mathbf{p})$*.*

Lemma

Assume both $\mathbf{u}(\mathbf{p})$ *and* $\ell(\mathbf{p})$ *are continuous for all* $\mathbf{p} \in \Delta$*. Then a symmetric equilibrium strategy* $\mathbf{p}^e \in \Delta$ *exists, and this strategy satisfies* $\mathbf{p}^e = \mathbf{f}(\mathbf{p}^e)$*.*

Deterministic fixed point algorithm

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})} \|\mathbf{p}-\mathbf{q}\|^2 \right\}$$

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

$$\arg\max_{\mathbf{q}\in\mathbb{R}^k} \phi(\mathbf{q};\mathbf{p}) = \mathbf{p} + \ell(\mathbf{p})\mathbf{u}(\mathbf{p})$$

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{\mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2\right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

$$\arg\max_{\mathbf{q}\in\mathbb{R}^k} \phi(\mathbf{q};\mathbf{p}) = \mathbf{p} + \underbrace{\ell(\mathbf{p})\mathbf{u}(\mathbf{p})}_{=:\mathbf{g}(\mathbf{p})}$$

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

$$\arg\max_{\mathbf{q}\in\mathbb{R}^k} \phi(\mathbf{q};\mathbf{p}) = \mathbf{p} + \underbrace{\ell(\mathbf{p})\mathbf{u}(\mathbf{p})}_{=:\mathbf{g}(\mathbf{p})}$$

The following iterative scheme

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

$$\arg\max_{\mathbf{q}\in\mathbb{R}^k} \phi(\mathbf{q};\mathbf{p}) = \mathbf{p} + \underbrace{\ell(\mathbf{p})\mathbf{u}(\mathbf{p})}_{=:\mathbf{g}(\mathbf{p})}$$

The following iterative scheme

$$\mathbf{p} \leftarrow \pi_\Delta\left(\mathbf{p} + \epsilon\mathbf{g}(\mathbf{p})\right)$$

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})}\|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

$$\arg\max_{\mathbf{q}\in\mathbb{R}^k} \phi(\mathbf{q};\mathbf{p}) = \mathbf{p} + \underbrace{\ell(\mathbf{p})\mathbf{u}(\mathbf{p})}_{=:\mathbf{g}(\mathbf{p})}$$

The following iterative scheme

$$\mathbf{p} \leftarrow \pi_\Delta\left(\mathbf{p} + \epsilon\mathbf{g}(\mathbf{p})\right)$$

converges to equilibrium,

Deterministic fixed point algorithm

Recall that

$$\mathbf{f}(\mathbf{p}) = \arg\max_{\mathbf{q}\in\Delta} \underbrace{\left\{ \mathbf{u}(\mathbf{p})'\mathbf{q} - \frac{1}{2\ell(\mathbf{p})} \|\mathbf{p}-\mathbf{q}\|^2 \right\}}_{=\phi(\mathbf{q};\mathbf{p})}$$

So the first order condition can be written as

$$\arg\max_{\mathbf{q}\in\mathbb{R}^k} \phi(\mathbf{q};\mathbf{p}) = \mathbf{p} + \underbrace{\ell(\mathbf{p})\mathbf{u}(\mathbf{p})}_{=:\mathbf{g}(\mathbf{p})}$$

The following iterative scheme

$$\mathbf{p} \leftarrow \pi_\Delta\left(\mathbf{p} + \epsilon\mathbf{g}(\mathbf{p})\right)$$

converges to equilibrium, where $\epsilon > 0$ and π_Δ is a projection onto Δ.

Stochastic fixed point algorithm

Stochastic fixed point algorithm

For an arbitrary initial strategy $\mathbf{p}^{(0)} \in \Delta$ the SA iteration is as follows:

Stochastic fixed point algorithm

For an arbitrary initial strategy $\mathbf{p}^{(0)} \in \Delta$ the SA iteration is as follows:

$$\mathbf{p}^{(n+1)} = \pi_{\Delta}\left(\mathbf{p}^{(n)} + \gamma_n \mathbf{G}^{(n)}\right),\ n \geq 0 \ ,$$

Stochastic fixed point algorithm

For an arbitrary initial strategy $\mathbf{p}^{(0)} \in \Delta$ the SA iteration is as follows:

$$\mathbf{p}^{(n+1)} = \pi_{\Delta}\left(\mathbf{p}^{(n)} + \gamma_n \mathbf{G}^{(n)}\right),\ n \geq 0 \ ,$$

where $\{\gamma_n\}_{n\geq 1}$ is a real positive sequence and $\mathbf{G}^{(n)}$ is an estimator for $\mathbf{g}(\mathbf{p}^{(n)})$.

Stochastic fixed point algorithm

For an arbitrary initial strategy $\mathbf{p}^{(0)} \in \Delta$ the SA iteration is as follows:

$$\mathbf{p}^{(n+1)} = \pi_{\Delta}\left(\mathbf{p}^{(n)} + \gamma_n \mathbf{G}^{(n)}\right),\ n \geq 0 ,$$

where $\{\gamma_n\}_{n\geq 1}$ is a real positive sequence and $\mathbf{G}^{(n)}$ is an estimator for $\mathbf{g}(\mathbf{p}^{(n)})$.

Challenge: $\mathbf{G}^{(n)}$ has to be unbiased for $\mathbf{g}(\mathbf{p}^{(n)})$!

Stochastic fixed point algorithm

For an arbitrary initial strategy $\mathbf{p}^{(0)} \in \Delta$ the SA iteration is as follows:

$$\mathbf{p}^{(n+1)} = \pi_\Delta \left(\mathbf{p}^{(n)} + \gamma_n \mathbf{G}^{(n)} \right), \ n \geq 0 \ ,$$

where $\{\gamma_n\}_{n \geq 1}$ is a real positive sequence and $\mathbf{G}^{(n)}$ is an estimator for $\mathbf{g}(\mathbf{p}^{(n)})$.

Challenge: $\mathbf{G}^{(n)}$ has to be unbiased for $\mathbf{g}(\mathbf{p}^{(n)})$!

Solution: We obtain unbiased estimators by simulating regenerative cycles.

Unbiased utility estimation

Unbiased utility estimation

Given $\mathbf{p}$,

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival.

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \overline{\mathbf{v}}(X_j, Y, \mathbf{p})$$

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \bar{\mathbf{v}}(X_j, Y, \mathbf{p})$$

When possible, it is more convenient to work with

$$\bar{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \,\big|\, X \right]$$

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \bar{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\bar{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \,\middle|\, X \right]$$

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \overline{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\overline{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \,\big|\, X \right]$$

Lemma

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \overline{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\overline{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \;\middle|\; X \right]$$

Lemma

Suppose for all $\mathbf{p} \in \Delta$,

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \overline{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\overline{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \,\middle|\, X \right]$$

Lemma

Suppose for all $\mathbf{p} \in \Delta$, $\ell^2(\mathbf{p}) < \infty$, and $\mathbf{v}$ is integrable.

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \overline{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\overline{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \;\middle|\; X \right]$$

Lemma

Suppose for all $\mathbf{p} \in \Delta$, $\ell^2(\mathbf{p}) < \infty$, *and* $\mathbf{v}$ *is integrable.*
Then $\mathrm{E}_{\mathbf{p}} \mathbf{G} = \mathbf{g}(\mathbf{p})$.

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \bar{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\bar{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \,\big|\, X \right]$$

Lemma

Suppose for all $\mathbf{p} \in \Delta$, $\ell^2(\mathbf{p}) < \infty$, *and* $\mathbf{v}$ *is integrable. Then* $\mathrm{E}_{\mathbf{p}}\mathbf{G} = \mathbf{g}(\mathbf{p})$ *(where* $\mathbf{g}(\mathbf{p}) = \ell(\mathbf{p})\mathbf{u}(\mathbf{p})$*).*

Unbiased utility estimation

Given $\mathbf{p}$, record $X_1, \ldots, X_L$, where X_j is the state realization at the j'th arrival. construct:

$$\mathbf{G} = \sum_{j=1}^{L} \overline{\mathbf{v}}(X_j)$$

When possible, it is more convenient to work with

$$\overline{\mathbf{v}}(X) = \mathrm{E}_Y \left[\mathbf{v}\big(X, Y, \mathbf{p}\big) \,\big|\, X \right]$$

Lemma

Suppose for all $\mathbf{p} \in \Delta$, $\ell^2(\mathbf{p}) < \infty$, *and* $\mathbf{v}$ *is integrable.*
Then $\mathrm{E}_{\mathbf{p}}\mathbf{G} = \mathbf{g}(\mathbf{p})$ *(where* $\mathbf{g}(\mathbf{p}) = \ell(\mathbf{p})\mathbf{u}(\mathbf{p})$*).*

Remark: In contrast, a naive sample-average estimator for $\mathbf{u}(\mathbf{p})$ is in general biased!

Convergence of SA algorithm

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}}\|\mathbf{G}\|^2 < \infty$.

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}} \|\mathbf{G}\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n \geq 1}$ satisfies

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}}\|\mathbf{G}\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n\geq 1}$ satisfies

$$\sum_{n=1}^{\infty} \gamma_n = \infty, \quad \sum_{n=1}^{\infty} \gamma_n^2 < \infty.$$

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}}\|\mathbf{G}\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n\geq 1}$ satisfies

$$\sum_{n=1}^{\infty} \gamma_n = \infty, \quad \sum_{n=1}^{\infty} \gamma_n^2 < \infty.$$

Theorem

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}} \|\mathbf{G}\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n \geq 1}$ satisfies

$$\sum_{n=1}^{\infty} \gamma_n = \infty, \quad \sum_{n=1}^{\infty} \gamma_n^2 < \infty.$$

Theorem

Suppose Assumptions A1-A4 are satisfied.

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}} \|\mathbf{G}\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n \geq 1}$ satisfies

$$\sum_{n=1}^{\infty} \gamma_n = \infty, \quad \sum_{n=1}^{\infty} \gamma_n^2 < \infty.$$

Theorem

Suppose Assumptions A1-A4 are satisfied. Then $\mathbf{p}^{(n)} \to_{\mathrm{as}} \mathbf{p}^e$ *as* $n \to \infty$,

Convergence of SA algorithm

For all $\mathbf{p} \in \Delta$:

Assumption A1: $\ell(\mathbf{p})$ is continuous with $\ell^2(\mathbf{p}) < \infty$.

Assumption A2: $\mathbf{u}(\mathbf{p})$ is integrable and continuous.

Assumption A3: $\mathrm{E}_{\mathbf{p}}\|\mathbf{G}\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n\geq 1}$ satisfies

$$\sum_{n=1}^{\infty} \gamma_n = \infty, \quad \sum_{n=1}^{\infty} \gamma_n^2 < \infty.$$

Theorem

Suppose Assumptions A1-A4 are satisfied. Then $\mathbf{p}^{(n)} \rightarrow_{\mathrm{as}} \mathbf{p}^e$ *as* $n \to \infty$*, such that* $\mathbf{f}(\mathbf{p}^e) = \mathbf{p}^e$*.*

Overview of applications

Overview of applications

We verify the convergence conditions and implement the algorithm for several unobservable queueing games:

Overview of applications

We verify the convergence conditions and implement the algorithm for several unobservable queueing games:

GI/G/1 in parallel (extending the motivating example).

Overview of applications

We verify the convergence conditions and implement the algorithm for several unobservable queueing games:

GI/G/1 in parallel (extending the motivating example).

Supermarket game: customers choose how many queues to inspect and join the shortest.

Overview of applications

We verify the convergence conditions and implement the algorithm for several unobservable queueing games:

- GI/G/1 in parallel (extending the motivating example).
- Supermarket game: customers choose how many queues to inspect and join the shortest.
- Sensing a finite buffer queue with an infinite buffer alternative.

Overview of applications

We verify the convergence conditions and implement the algorithm for several unobservable queueing games:

- GI/G/1 in parallel (extending the motivating example).
- Supermarket game: customers choose how many queues to inspect and join the shortest.
- Sensing a finite buffer queue with an infinite buffer alternative.

The above are all systems with no explicit stationary solution.

Overview of applications

We verify the convergence conditions and implement the algorithm for several unobservable queueing games:

- GI/G/1 in parallel (extending the motivating example).
- Supermarket game: customers choose how many queues to inspect and join the shortest.
- Sensing a finite buffer queue with an infinite buffer alternative.

The above are all systems with no explicit stationary solution.

The algorithm is easily implemented (even without verification of the conditions).

Application: Hassin & S. 2017

Application: Hassin & S. 2017

One $\mathrm{M/G/1/1}$ queue (1) and one $\mathrm{M}(s)\mathrm{/G/1}$ queue (2)

Application: Hassin & S. 2017

One M/G/1/1 queue (1) and one $\mathrm{M}(s)$/G/1 queue (2)

Delay-sensitive customers can choose to make a *costly* attempt to join queue 1 (a_1),

Application: Hassin & S. 2017

One M/G/1/1 queue (1) and one $\mathrm{M}(s)$/G/1 queue (2)

Delay-sensitive customers can choose to make a *costly* attempt to join queue 1 (a_1), if failed they're rerouted to queue 2.

Application: Hassin & S. 2017

One M/G/1/1 queue (1) and one $\mathrm{M}(s)$/G/1 queue (2)

Delay-sensitive customers can choose to make a *costly* attempt to join queue 1 (a_1), if failed they're rerouted to queue 2.

Their alternative (a_2) is to join queue 2 directly.

Application: Hassin & S. 2017

One $\mathrm{M/G/1/1}$ queue (1) and one $\mathrm{M}(s)/\mathrm{G}/1$ queue (2)

Delay-sensitive customers can choose to make a *costly* attempt to join queue 1 (a_1), if failed they're rerouted to queue 2.

Their alternative (a_2) is to join queue 2 directly.

$\Rightarrow$ The effective arrival process to queue 2 is not renewal.

Application: Hassin & S. 2017

One M/G/1/1 queue (1) and one $\mathrm{M}(s)/\mathrm{G}/1$ queue (2)

Delay-sensitive customers can choose to make a *costly* attempt to join queue 1 (a_1), if failed they're rerouted to queue 2.

Their alternative (a_2) is to join queue 2 directly.

$\Rightarrow$ The effective arrival process to queue 2 is not renewal.

For exponential services, the (unique) equilibrium can be approached numerically.

Application: Hassin & S. 2017

One $\mathrm{M}/\mathrm{G}/1/1$ queue (1) and one $\mathrm{M}(s)/\mathrm{G}/1$ queue (2)

Delay-sensitive customers can choose to make a *costly* attempt to join queue 1 (a_1), if failed they're rerouted to queue 2.

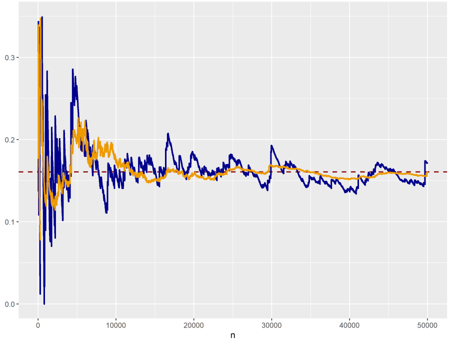
Their alternative (a_2) is to join queue 2 directly.

$\Rightarrow$ The effective arrival process to queue 2 is not renewal.

For exponential services, the (unique) equilibrium can be approached numerically.

We implement variance reduction techniques (using control variates) and dynamic step size selection in the SA algorithm.

Application: Hassin & S. 2017



$p_1^{(n)}$ is plotted vs. n. Blue curve corresponds with crude implementation, orange with the refined version. Red dashed line depicts correct equilibrium.

Extensions and refinements

Extensions and refinements

Variance reduction techniques can be applied to make the algorithm more efficient.

Extensions and refinements

Variance reduction techniques can be applied to make the algorithm more efficient.

In some cases we can relax the assumption of system stability on all of the strategy space (if we know some properties of the stability region).

Extensions and refinements

Variance reduction techniques can be applied to make the algorithm more efficient.

In some cases we can relax the assumption of system stability on all of the strategy space (if we know some properties of the stability region).

The algorithm can be modified to derive socially optimal strategies.

Extensions and refinements

Variance reduction techniques can be applied to make the algorithm more efficient.

In some cases we can relax the assumption of system stability on all of the strategy space (if we know some properties of the stability region).

The algorithm can be modified to derive socially optimal strategies.

An interesting challenge would be to allow more frequent updating of the strategy during the simulation.

Summary of results

Summary of results

We introduce:

Summary of results

We introduce:

> SA algorithm for computing SNE in a general unobservable queueing game

Summary of results

We introduce:

- SA algorithm for computing SNE in a general unobservable queueing game
- Unbiased estimation of total utility observed during a cycle

Summary of results

We introduce:

SA algorithm for computing SNE in a general unobservable queueing game

Unbiased estimation of total utility observed during a cycle

Verifiable conditions for almost sure convergence

Summary of results

We introduce:

- SA algorithm for computing SNE in a general unobservable queueing game
- Unbiased estimation of total utility observed during a cycle
- Verifiable conditions for almost sure convergence

The algorithm is practical, extendable, and easy to implement using simulation.

Questions?

Thank you!