Stochastic approximation of symmetric Nash equilibria in queueing games

Ran I. Snitkovsky\textsuperscript{1,2}

coa-authored with

Liron Ravner \textsuperscript{3}

\textsuperscript{1}SRIBD, CUHK Shenzhen
\textsuperscript{2}Columbia Business School
\textsuperscript{3}Department of Statistics, University of Haifa

2021
Symmetric unobservable queueing games

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1’s
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up
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 ⇒ 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 ⇒ Takes action ⇒ Undergoes processing ⇒ 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 ⇒ Takes action ⇒ Undergoes processing ⇒ Receives utility ⇒ 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 ⇒ Takes action ⇒ Undergoes processing ⇒ Receives utility ⇒ 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
Stochastic approximation of symmetric Nash equilibria in queueing games

Ran Snitkovsky

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1's
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up

Symmetric unobservable queueing games

Customer utility depends on:

Their action
The population strategy
(Steady-)state realization
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

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1’s
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up
Stochastic approximation of symmetric Nash equilibria in queueing games

Ran Snitkovsky

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1’s
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up

Unobservable M/M/1 Model

An M/M/1 system
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim$ Poisson($\lambda$)
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals \( \sim \) Poisson(\( \lambda \))

Possible actions:
Unobservable M/M/1 Model

An M/M/1 system

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

Possible actions: Join or Balk
Unobservable M/M/1 Model

An M/M/1 system

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

Possible actions: Join or Balk

A strategy $p = (p, 1 - p)$, with $p =$ probability of Join
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim$ Poisson($\lambda$)

Possible actions: Join or Balk

A strategy $p = (p, 1 - p)$, with $p =$ probability of Join

$$\text{util. from JOIN} = R + C \times \text{waiting time}$$
Unobservable M/M/1 Model

An M/M/1 system

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

Possible actions: Join or Balk

A strategy \( p = (p, 1 - p) \), with \( p \) = probability of Join

\[
\text{util. from JOIN} = R + C \times \text{waiting time}
\]

\[
\text{util. from BALK} = 0
\]
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim$ Poisson($\lambda$)

Possible actions: Join or Balk

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

\[
\text{util. from JOIN} = R + C \times \text{waiting time}
\]
\[
\text{util. from BALK} = 0
\]

Customers wish to maximize expected steady-state utility.
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals \( \sim \) Poisson(\( \lambda \))

Possible actions: Join or Balk

A strategy \( p = (p, 1 - p) \), with \( p \) = probability of Join

\[
\begin{align*}
u_1(p) &= R + C \times \text{waiting time} \\
\text{util. from BALK} &= 0
\end{align*}
\]

Customers wish to maximize expected steady-state utility.
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim$ Poisson($\lambda$)

Possible actions: Join or Balk

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

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

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 ∼ Poisson(λ)

Possible actions: Join or Balk

A strategy \( p = (p, 1 - p) \), with \( p \) = probability of Join

\[
\begin{align*}
u_1(p) &= R + C \times E_p \left( \text{waiting time} \right) \\
u_2(p) &= 0
\end{align*}
\]

Customers wish to maximize expected steady-state utility.
Unobservable M/M/1 Model

An M/M/1 system

Customers inter-arrivals $\sim$ Poisson($\lambda$)

Possible actions: Join or Balk

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

\[
u_1(\mathbf{p}) = R + C \times E_p \left( \frac{\text{waiting time}}{} \right)
\]

\[
u_2(\mathbf{p}) = 0
\]

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 $p^e = (p^e, 1 - p^e)$
Unobservable M/M/1 Model

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

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

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

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

proj. of $\mathcal{BR}(\mathbf{p})$ onto $[0, 1]$
Unobservable M/M/1 Model

In this model $p_e$ is available in closed form, solving:
Unobservable M/M/1 Model

In this model \( p^e \) is available in closed form, solving:

\[
u_1(p) = u_2(p)\]

...and similarly, for unobservable M/G/1 and G/M/1 models.
In this model $p^e$ is available in closed form, solving:

$$u_1(p) = u_2(p) = 0$$
Unobservable M/M/1 Model

In this model $p_e$ is available in closed form, solving:

$$u_1(p) = u_2(p) = 0$$

...and similarly, for unobservable M/G/1 and G/M/1 models.
In this model $p^e$ is available in closed form, solving:

$$u_1(p) = u_2(p) = 0$$

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

What about a GI/G/1 queue?
Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available. BR is not (lower hemi-)continuous. Lengthy simulations to verify SNE conditions for many p's are impracticable. Our goal: Find a SNE strategy by running a single simulation of the system, with dynamic updating of the strategy.
Challenges and objective

For non-elementary queueing processes the steady-state distribution is not available.
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 \( p \)'s are impracticable.
Challenges and objective

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

\[ \mathcal{BR} \text{ is not (lower hemi-)continuous.} \]

Lengthy simulations to verify SNE conditions for many \( p \)'s are impracticable.

Our goal:
Challenges and objective

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

\[ \mathcal{BR} \text{ is not (lower hemi-)continuous.} \]

Lengthy simulations to verify SNE conditions for many \( p \)'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 are two parallel queues with heterogeneous service time distributions: $Y_1 \sim F_1$ and $Y_2 \sim F_2$, with means $E[Y_1] = \mu_1 \geq E[Y_2] = \mu_2$. Independent inter-arrival time distribution $H$ with mean $\lambda$, 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

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 $E_{Y_1} = \mu_1 \geq E_{Y_2} = \mu_2$, 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

\[
E[Y_1] = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = E[Y_2]
\]
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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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

\[
EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2
\]

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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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

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

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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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

$$EY_1 = \frac{1}{\mu_1} \geq \frac{1}{\mu_2} = EY_2$$

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:

\[ A = \{ \text{Join 1, Join 2, Balk} \} \]
Parallel GI/G/1 queues

Denote the action set:

\[ A = \{ a_1, a_2, a_3 \} \]
Parallel GI/G/1 queues

Denote the action set:

\[ A = \{ a_1, a_2, a_3 \} \]

A strategy is a distribution \( p = (p_1, p_2, p_3) \) over \( A \).
Parallel GI/G/1 queues

Denote the action set:

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

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

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

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

$$\text{If } EY^2_m < \infty, \text{ then } w_m(\mathbf{p}_m) := E_{\mathbf{p}_m} W_m < \infty.$$
Parallel GI/G/1 queues

Denote the action set:

\[ \mathcal{A} = \{ a_1, a_2, a_3 \} \]

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:

\[ A = \{ a_1, a_2, a_3 \} \]

A strategy is a distribution \( p = (p_1, p_2, p_3) \) over \( A \).

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:

\[ \mathcal{A} = \{ a_1, a_2, a_3 \} \]

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 \( \mathbb{E} Y^2_m < \infty \),
Parallel GI/G/1 queues

Denote the action set:

\[ A = \{ a_1, a_2, a_3 \} \]

A strategy is a distribution \( p = (p_1, p_2, p_3) \) over \( A \).

For queue \( m \in \{1, 2\} \):

- If \( \lambda p_m < \mu_m \), then the stationary workload \( W_m \) exists.
- If \( EY_m^2 < \infty \), then \( w_m(p_m) := E_p W_m < \infty \).
Parallel GI/G/1 queues

The SNE condition:
Parallel GI/G/1 queues

The SNE condition:

\[ p^e \in BR(p^e) \]
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:

\[ p^e \in B\mathcal{R}(p^e) = \arg\max_{p \in \Delta} p'u(p^e) \]

where

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

It can be verified that \( p^e \) exists uniquely. However, an expression for \( w_m(p_m) \) is not available.

How to compute \( p^e \)?
Parallel GI/G/1 queues

The SNE condition:

\[ p^e \in B(\mathcal{R}(p^e) = \arg\max_{p \in \Delta} p' u(p^e) \]

where

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

It can be verified that \( p^e \) exists uniquely. However, an expression for \( w_m(p_m) \) is not available. How to compute \( p^e \)?
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

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

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:

\[ p^e \in BR(p^e) = \arg \max_{p \in \Delta} p'u(p^e) \]

where

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

It can be verified that \( p_e \) exists uniquely.

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

How to compute \( p_e \)?
SA Algorithm

We suggest a simulation-based, SA (Robbins-Monro) algorithm. The algorithm involves a regeneration cycle, which is the time between two arrival instants to an empty system. The cycle length is the number of arrivals during a cycle (including balkings). Our stability assumptions imply that the cycle length is finite (a.s.).
SA Algorithm

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

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

Regeneration cycle:
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:**
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).
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.
At iteration $n \geq 1$ of the algorithm:

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

Let $L$ denote the cycle length.
SA Algorithm

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

- Given a strategy $p^{(n)}$, generate 1 cycle.
- Let $L$ denote the cycle length.
- Record the vector total expected utilities:

$$G(n) = L \sum_{j=1}^{m} \begin{pmatrix} \sum_{i=1}^{m} (R_{i} - C_{i} \cdot (X_{i}[1] + 1/\mu_{i})) \\ \sum_{i=1}^{m} (R_{i} - C_{i} \cdot (X_{i}[2] + 1/\mu_{i})) \end{pmatrix},$$

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

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

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

Let $L$ denote the cycle length.

Record the vector total expected utilities:

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

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

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

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

Let $L$ denote the cycle length.

Record the vector total expected utilities:

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

where $X_j^m$ is the workload in queue $m = 1, 2$ at the $j$'th arrival.
SA Algorithm

Start with an arbitrary strategy $p^{(0)}$, and initial step size $\gamma_0 > 0$. Update the strategy as follows:

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

projecting onto $\Delta$ when necessary. It can be shown that $p^{(n)} \rightarrow p^e$. 

---
SA Algorithm

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

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

Update the strategy as follows:
SA Algorithm

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

Update the strategy as follows:

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

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

Update the strategy as follows:

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

projecting onto $\Delta$ 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 $\Delta$ when necessary.

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

\[ F_1 \sim \text{Beta}(10, 10) + 0.5, \]
Simulation Results

\[ F_1 \sim \text{Beta}(10, 10) + 0.5, \quad F_2 \sim \text{Bernoulli}(0.1) \cdot 10, \]
Simulation Results

\[ F_1 \sim \text{Beta}(10, 10) + 0.5, \ F_2 \sim \text{Bernoulli}(0.1) \cdot 10, \ \text{and} \ \ H \sim \text{Gamma}(0.1, 11) \]
Simulation Results

\[ F_1 \sim \text{Beta}(10, 10) + 0.5, \ F_2 \sim \text{Bernoulli}(0.1) \cdot 10, \ \text{and} \ \ H \sim \text{Gamma}(0.1, 11) \ \text{with} \ R = 5, \ C = 1 \]
Simulation Results

\[ F_1 \sim \text{Beta}(10, 10) + 0.5, \quad F_2 \sim \text{Bernoulli}(0.1) \cdot 10, \quad \text{and} \quad H \sim \text{Gamma}(0.1, 11) \] 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).

\[ \varepsilon \text{-equilibrium condition satisfied for } \varepsilon < 0.02 \text{ with } >0.99 \text{ certainty} \]
Simulation Results

\[ F_1 \sim \text{Beta}(10, 10) + 0.5, \quad F_2 \sim \text{Bernoulli}(0.1) \cdot 10, \quad \text{and} \quad H \sim \text{Gamma}(0.1, 11) \text{ with } R = 5, \quad C = 1 \]

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

\( \varepsilon \)-equilibrium condition satisfied for \( \varepsilon < 0.02 \) with \( > 0.99 \) certainty
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 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.
The Robbins-Monro algorithm

Goal:
**The Robbins-Monro algorithm**

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

The Robbins-Monro algorithm 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.
The Robbins-Monro algorithm

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

**Iterative solution:**
The Robbins-Monro algorithm

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

**Iterative solution:** Given a sequence $\{\gamma_n\}$ of positive step sizes, perform:
The Robbins-Monro algorithm

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

**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 \( g : \mathbb{R} \rightarrow \mathbb{R} \).

**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 $g : \mathbb{R} \rightarrow \mathbb{R}$.

**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 $g : \mathbb{R} \rightarrow \mathbb{R}$.

**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 $g : \mathbb{R} \rightarrow \mathbb{R}$.

**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 \( g : \mathbb{R} \rightarrow \mathbb{R} \).

**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 \( g : \mathbb{R} \rightarrow \mathbb{R} \).

**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 $g : \mathbb{R} \rightarrow \mathbb{R}$.

**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

Literature about unobservable queueing games is extensive.
Relevant literature

Literature about unobservable queueing games is extensive.

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


**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:
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 Markovian 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 Markovian 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.
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 Markovian 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

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1's
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up
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:
Stochastic approximation of symmetric Nash equilibria in queueing games

Ran Snitkovsky

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1’s
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up

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: $\mathcal{A} = \{a_1, \ldots, a_k\}$.

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

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

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

Assume the system starts empty: $X_0(p) = 0$. Let $L(p) = \inf\{n \geq 1 : X_n(p) = 0\}$ be the cycle length. Assume $\ell(p) < \infty$ for any $p \in \Delta \Rightarrow$ there exists a stationary distribution, $X_n(p) \to_d X(p)$. Thus, the system is regenerative at 0 and is stable for any strategy.
Regenerative structure

Assume the system starts empty:
Regenerative structure

Assume the system starts empty: \( X_0(p) = 0 \).
Regenerative structure

Assume the system starts empty: $X_0(p) = 0$.

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

Assume the system starts empty: \( X_0(p) = 0. \)

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

Let \( \ell^k(p) = \mathbb{E}_p L^k(p). \)
Regenerative structure

Assume the system starts empty: $X_0(p) = 0$.

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

Let $\ell^k(p) = E_p L^k(p)$.

Assume $\ell(p) < \infty$ for any $p \in \Delta$
Regenerative structure

Assume the system starts empty: $X_0(p) = 0$.

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

Let $\ell^k(p) = E_p L^k(p)$.

Assume $\ell(p) < \infty$ for any $p \in \Delta \Rightarrow$ there exists a stationary distribution, $X_n(p) \rightarrow_d X(p)$. 

Assume the system starts empty: $X_0(p) = 0$.

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

Let $\ell^k(p) = E_p L^k(p)$.

Assume $\ell(p) < \infty$ for any $p \in \Delta \Rightarrow$ there exists a stationary distribution, $X_n(p) \rightarrow_d X(p)$.
Assume the system starts empty: $X_0(p) = 0$.

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

Let $\ell^k(p) = E_p L^k(p)$.

Assume $\ell(p) < \infty$ for any $p \in \Delta$ $\Rightarrow$ there exists a stationary distribution, $X_n(p) \rightarrow_d X(p)$.

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

The utility of a customer choosing action \( a_i \) is \( v_i(x, y, p) \), where:
- \( x \) is the (realized) state upon arrival,
- \( y \) is some realized random outcome (e.g., service time),
- \( p \) is the population strategy.

Let \( v_i(x, y, p) = (v_i(x, y, p))_{i=1}^k \).

Each coordinate \( i \) of \( u(p) \) corresponds to the mean stationary utility from action \( a_i \).
Utility structure

The utility of a customer choosing action $a_i$ is $v_i(x, y, p)$,
Utility structure

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

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

Let $v(x, y, p) = (v_i(x, y, p))_{i=1}^k$.
Each coordinate $i$ of $v(p)$ corresponds to the mean stationary utility from action $a_i$. 
Utility structure

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

- $x$ is the (realized) state upon arrival,
Utility structure

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

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

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

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

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

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

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


Utility structure

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

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

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

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

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

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

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

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

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

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

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

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

The mean stationary utility vector is
Utility structure

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

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

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

The mean stationary utility vector is

\[
u(p) = \mathbb{E}_p \left[ v(X(p), Y, p) \right].\]
Utility structure

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

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

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

The mean stationary utility vector is

$$u(p) = \mathbb{E}_p \left[ v(X(p), Y, p) \right].$$

Each coordinate $i$ of $u(p)$ corresponds to the mean stationary utility from action $a_i$. 
Symmetric Nash equilibrium

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

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

$$\text{Solution: } \text{We use a surrogate best response function.}$$
Symmetric Nash equilibrium

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

$$p \in \arg \max_{q \in \Delta} u(p)'q$$
A Symmetric Nash Equilibrium strategy is a strategy $p \in \Delta$ such that

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

$$=: BR(p)$$
Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy \( p \in \Delta \) such that

\[
p \in \arg \max_{q \in \Delta} u(p)' q
\]

Problem:

\[
= \mathcal{BR}(p)
\]
A Symmetric Nash Equilibrium strategy is a strategy \( p \in \Delta \) such that

\[
p \in \arg \max_{q \in \Delta} u(p)'q
\]

\[=: BR(p)\]

**Problem:** \( BR(p) \) is a set-valued mapping, and not lower-hemicontinuous.
A Symmetric Nash Equilibrium strategy is a strategy $p \in \Delta$ such that

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

$$=:BR(p)$$

**Problem:** $BR(p)$ is a set-valued mapping, and not lower-hemicontinuous.

**Solution:**
Symmetric Nash equilibrium

A Symmetric Nash Equilibrium strategy is a strategy \( p \in \Delta \) such that

\[
p \in \arg \max_{q \in \Delta} u(p)'q
\]

\[=:BR(p)\]

**Problem:** \( BR(p) \) is a set-valued mapping, and not lower-hemicontinuous.

**Solution:** We use a surrogate best response function.
Best response surrogate

For a vector $q \in \mathbb{R}^k$ and a strategy $p \in \Delta$, define the function $\phi$: $\mathbb{R}^k \times \Delta \rightarrow \mathbb{R}$ as $\phi(q; p) = u(p)'q - \frac{1}{2} \ell(p) \| p - q \|^2$.

This yields a surrogate best response function: $f(p) = \arg \max_{q \in \Delta} \phi(q; p) = \arg \max_{q \in \Delta} \{ u(p)'q - \frac{1}{2} \ell(p) \| p - q \|^2 \}$.

The function $f$ fixes the discontinuities in BR. The choice of $\ell(p)$ will be made clear later.
Best response surrogate

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

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

This yields a surrogate best response function:

\[ f(\mathbf{p}) = \text{arg max}_{\mathbf{q} \in \Delta} \phi(\mathbf{q}; \mathbf{p}) = \text{arg max}_{\mathbf{q} \in \Delta} \left\{ u(\mathbf{p})' \mathbf{q} - \frac{1}{2} \ell(\mathbf{p}) \| \mathbf{p} - \mathbf{q} \|^2 \right\}. \]

The function $f$ fixes the discontinuities in $\text{BR}^!$. The choice of $\ell(\mathbf{p})$ will be made clear later.
Best response surrogate

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

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

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

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

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

This yields a surrogate best response function:
Best response surrogate

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

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

This yields a surrogate best response function:

\[
f(p) = \arg \max_{q \in \Delta} \phi(q; p)
\]
Best response surrogate

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

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

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

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

The function $f$ fixes the discontinuities in $BR$!
Best response surrogate

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

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

The function $f$ fixes the discontinuities in $BR$!

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

Lemma
Lemma

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

**Lemma**

\[ \text{A strategy } p \in \Delta \text{ is a Symmetric Nash Equilibrium, i.e., } p \in BR(p), \text{ if and only if it satisfies } p = f(p). \]
Alternative equilibrium condition

**Lemma**

A strategy \( p \in \Delta \) is a Symmetric Nash Equilibrium, i.e., \( p \in BR(p) \), if and only if it satisfies \( p = f(p) \).
Lemma

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

Lemma

Assume both $u(p)$ and $\ell(p)$ are continuous for all $p \in \Delta$. 
Alternative equilibrium condition

Lemma
A strategy \( p \in \Delta \) is a Symmetric Nash Equilibrium, i.e., \( p \in BR(p) \), if and only if it satisfies \( p = f(p) \).

Lemma
Assume both \( u(p) \) and \( \ell(p) \) are continuous for all \( p \in \Delta \). Then a symmetric equilibrium strategy \( p^e \in \Delta \) exists, and this strategy satisfies \( p^e = f(p^e) \).
Deterministic fixed point algorithm

Recall that So the first order condition can be written as
The following iterative scheme
\[ p \leftarrow \pi \Delta \left( p + \epsilon g(p) \right) \]
converges to equilibrium, where \( \epsilon > 0 \) and \( \pi \Delta \) is a projection onto \( \Delta \).
Deterministic fixed point algorithm

Recall that

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

Recall that

\[
f(p) = \arg \max_{q \in \Delta} \left\{ u(p)'q - \frac{1}{2\ell(p)}\|p - q\|^2 \right\} = \phi(q;p)
\]
Deterministic fixed point algorithm

Recall that

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

So the first order condition can be written as
Deterministic fixed point algorithm

Recall that

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

So the first order condition can be written as

\[ \arg \max_{q \in \mathbb{R}^k} \phi(q; p) = p + \ell(p)u(p) \]
Deterministic fixed point algorithm

Recall that

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

So the first order condition can be written as

$$\arg\max_{q \in \mathbb{R}^k} \phi(q;p) = p + \ell(p)u(p) =: g(p)$$
Deterministic fixed point algorithm

Recall that

\[ f(p) = \underset{q \in \Delta}{\arg \max} \left\{ u(p)'q - \frac{1}{2\ell(p)}\|p - q\|^2 \right\} = \phi(q; p) \]

So the first order condition can be written as

\[ \underset{q \in \mathbb{R}^k}{\arg \max} \phi(q; p) = p + \ell(p)u(p) =: g(p) \]

The following iterative scheme

\[ p \leftarrow \pi_{\Delta}(p + \epsilon g(p)) \]

converges to equilibrium, where \( \epsilon > 0 \) and \( \pi_{\Delta} \) is a projection onto \( \Delta \).
Deterministic fixed point algorithm

Recall that

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

So the first order condition can be written as

\[ \arg \max_{q \in \mathbb{R}^k} \phi(q; p) = p + \ell(p)u(p) =: g(p) \]

The following iterative scheme

\[ p \leftarrow \pi_{\Delta} \left( p + \epsilon g(p) \right) \]
Deterministic fixed point algorithm

Recall that

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

So the first order condition can be written as

\[ \arg \max_{q \in \mathbb{R}^k} \phi(q; p) = p + \ell(p)u(p) =: g(p) \]

The following iterative scheme

\[ p \leftarrow \pi_{\Delta} (p + \epsilon g(p)) \]

converges to equilibrium,
Deterministic fixed point algorithm

Recall that

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

So the first order condition can be written as

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

The following iterative scheme

\[
p \leftarrow \pi_{\Delta} (p + \epsilon g(p))
\]

converges to equilibrium, where \( \epsilon > 0 \) and \( \pi_{\Delta} \) is a projection onto \( \Delta \).
Stochastic fixed point algorithm

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

$$p(n+1) = \pi \Delta(p(n) + \gamma n G(n)),$$

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

Challenge: $G(n)$ has to be unbiased for $g(p(n))$.

Solution: We obtain unbiased estimators by simulating regenerative cycles.
Stochastic fixed point algorithm

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

$$p^{(n+1)} = \pi \Delta (p^{(n)} + \gamma^n g(n)),$$

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

Challenge: $g(n)$ has to be unbiased for $g(p^{(n)})$.

Solution: We obtain unbiased estimators by simulating regenerative cycles.
Stochastic fixed point algorithm

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

\[
p^{(n+1)} = \pi_{\Delta} \left( p^{(n)} + \gamma_n G^{(n)} \right), \quad n \geq 0,
\]
Stochastic fixed point algorithm

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

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

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

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

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

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

**Challenge:** $G^{(n)}$ has to be unbiased for $g(p^{(n)})$!
Stochastic fixed point algorithm

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

\[
p^{(n+1)} = \pi\Delta \left( p^{(n)} + \gamma_n G^{(n)} \right), \quad n \geq 0,
\]

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

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

**Solution:** We obtain unbiased estimators by simulating regenerative cycles.
Unbiased utility estimation

Given $p$, record $X_1, \ldots, X_L$, where $X_j$ is the state realization at the $j$'th arrival. When possible, it is more convenient to work with $v(X) = E_Y[v(X, Y, p)|X]$. Lemma Suppose for all $p \in \Delta$, $\ell^2(p) < \infty$, and $v$ is integrable. Then $E\mu = g(p)$ (where $g(p) = \ell(p)u(p)$). Remark: In contrast, a naive sample-average estimator for $u(p)$ is in general biased!
Unbiased utility estimation

Given $\mathbf{p}$,
Unbiased utility estimation

Given \( p \), record \( X_1, \ldots, X_L \), where \( X_j \) is the state realization at the \( j \)'th arrival.
Unbiased utility estimation

Given $p$, record $X_1, \ldots, X_L$, where $X_j$ is the state realization at the $j$’th arrival. construct:
Unbiased utility estimation

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

$$G = \sum_{j=1}^{L} v(X_j, Y, p)$$
Unbiased utility estimation

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

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

When possible, it is more convenient to work with

$$ \bar{v}(X) = \mathbb{E}_Y [v(X, Y, p) | X] $$
Unbiased utility estimation

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

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

When possible, it is more convenient to work with

$$\bar{v}(X) = \mathbb{E}_Y \left[ v(X, Y, p) \mid X \right]$$

Lemma

Suppose for all $p \in \Delta$, $\ell_2(p) < \infty$, and $v$ is integrable.

Then $\mathbb{E}_p G = g(p)$ (where $g(p) = \ell(p) u(p)$).

Remark:

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

\[
G = \sum_{j=1}^{L} \bar{v}(X_j)
\]

When possible, it is more convenient to work with

\[
\bar{v}(X) = \mathbb{E}_Y \left[ v(X, Y, p) \mid X \right]
\]

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:

\[
G = \sum_{j=1}^{L} \overline{v}(X_j)
\]

When possible, it is more convenient to work with

\[
\overline{v}(X) = \mathbb{E}_Y \left[ v(X, Y, \mathbf{p}) \mid X \right]
\]

Lemma

*Suppose for all \( \mathbf{p} \in \Delta \),*
Unbiased utility estimation

Given \( p \), record \( X_1, \ldots, X_L \), where \( X_j \) is the state realization at the \( j \)'th arrival. Construct:

\[
G = \sum_{j=1}^{L} \bar{v}(X_j)
\]

When possible, it is more convenient to work with

\[
\bar{v}(X) = \mathbb{E}_Y \left[ v(X, Y, p) \mid X \right]
\]

Lemma

Suppose for all \( p \in \Delta \), \( \ell^2(p) < \infty \), and \( v \) 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:

\[
G = \sum_{j=1}^{L} \bar{v}(X_j)
\]

When possible, it is more convenient to work with

\[
\bar{v}(X) = \mathbb{E}_Y \left[ v(X, Y, \mathbf{p}) \mid X \right]
\]

Lemma

Suppose for all \( \mathbf{p} \in \Delta \), \( \ell^2(\mathbf{p}) < \infty \), and \( v \) is integrable. Then \( \mathbb{E}_p G = g(\mathbf{p}) \).
Unbiased utility estimation

Given \( p \), record \( X_1, \ldots, X_L \), where \( X_j \) is the state realization at the \( j \)'th arrival. construct:

\[
G = \sum_{j=1}^{L} \bar{v}(X_j)
\]

When possible, it is more convenient to work with

\[
\bar{v}(X) = \mathbb{E}_Y [v(X, Y, p) \mid X]
\]

**Lemma**

*Suppose for all \( p \in \Delta, \ell^2(p) < \infty, \) and \( v \) is integrable. Then \( \mathbb{E}_p G = g(p) \) (where \( g(p) = \ell(p)u(p) \)).*
Unbiased utility estimation

Given \( p \), record \( X_1, \ldots, X_L \), where \( X_j \) is the state realization at the \( j \)'th arrival. Construct:

\[
G = \sum_{j=1}^{L} \bar{v}(X_j)
\]

When possible, it is more convenient to work with

\[
\bar{v}(X) = \mathbb{E}_Y \left[ v(X, Y, p) \mid X \right]
\]

Lemma

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

Then \( \mathbb{E}_p G = g(p) \) (where \( g(p) = \ell(p)u(p) \)).

Remark: In contrast, a naive sample-average estimator for \( u(p) \) is in general biased!
Convergence of SA algorithm

For all $p \in \Delta$:

Assumption A1: $\ell(p)$ is continuous with $\ell_2(p) < \infty$.

Assumption A2: $u(p)$ is integrable and continuous.

Assumption A3: $E_p \|G\|^2 < \infty$.

Assumption A4: The step-size sequence $\{\gamma_n\}_{n \geq 1}$ satisfies $\sum_{n=1}^{\infty} \gamma_n = \infty$, $\sum_{n=1}^{\infty} \gamma_n^2 < \infty$.

Theorem

Suppose Assumptions A1-A4 are satisfied. Then $p_n \rightarrow p_e$ as $n \rightarrow \infty$, such that $f(p_e) = p_e$. 

Implementation

Applications

H-S(2017) model

Conclusion

Extensions

Wrap-up
Stochastic approximation of symmetric Nash equilibria in queueing games

Ran Snitkovsky

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1's
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up

Convergence of SA algorithm

For all \( p \in \Delta \):
Convergence of SA algorithm

For all \( p \in \Delta: \)

**Assumption A1:** \( \ell(p) \) is continuous with \( \ell^2(p) < \infty. \)
Convergence of SA algorithm

For all $p \in \Delta$:

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

**Assumption A2:** $u(p)$ is integrable and continuous.
For all $p \in \Delta$:

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

**Assumption A2:** $u(p)$ is integrable and continuous.

**Assumption A3:** $\mathbb{E}_p \|G\|^2 < \infty$. 
Convergence of SA algorithm

For all $p \in \Delta$:

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

**Assumption A2:** $u(p)$ is integrable and continuous.

**Assumption A3:** $E_p \|G\|^2 < \infty$.

**Assumption A4:** The step-size sequence $\{\gamma_n\}_{n \geq 1}$ satisfies
Convergence of SA algorithm

For all $p \in \Delta$:

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

**Assumption A2:** $u(p)$ is integrable and continuous.

**Assumption A3:** $\mathbb{E}_p \|G\|^2 < \infty$.

**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 \( p \in \Delta \):

**Assumption A1:** \( \ell(p) \) is continuous with \( \ell^2(p) < \infty \).

**Assumption A2:** \( u(p) \) is integrable and continuous.

**Assumption A3:** \( E_p \|G\|^2 < \infty \).

**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 \( p \in \Delta \):

**Assumption A1:** \( \ell(p) \) is continuous with \( \ell^2(p) < \infty \).

**Assumption A2:** \( u(p) \) is integrable and continuous.

**Assumption A3:** \( \mathbb{E}_p \| G \| ^2 < \infty \).

**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 $p \in \Delta$:

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

**Assumption A2:** $u(p)$ is integrable and continuous.

**Assumption A3:** $E_p \|G\|^2 < \infty$.

**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

$$p^{(n)} \to p^e \text{ as } n \to \infty,$$
Convergence of SA algorithm

For all $p \in \Delta$:

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

**Assumption A2**: $u(p)$ is integrable and continuous.

**Assumption A3**: $E_p \|G\|^2 < \infty$.

**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 $p^{(n)} \rightarrow_{as} p^e$ as $n \rightarrow \infty$, such that $f(p^e) = p^e$. 

Stochastic approximation of symmetric Nash equilibria in queueing games

Ran Snitkovsky

Intro.
Queueing games
Unobservable M/M/1
Research purpose

Motiv. Example
Parallel GI/G/1’s
SA Simulation

SA and Queues
R-M Algo.
Literature

Gen. Framework
General formulation
Fixed-point algo.
SA scheme
Utility estimation
Convergence

Implementation
Applications
H-S(2017) model

Conclusion
Extensions
Wrap-up
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).
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

One M/G/1 queue (1) and one M/s/G/1 queue (2). Delay-sensitive customers can choose to make a costly attempt to join queue 1 (a₁), if failed they're rerouted to queue 2. Their alternative (a₂) is to join queue 2 directly. ⇒ 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

One M/G/1/1 queue (1) and one M(s)/G/1 queue (2)
Application: Hassin & S. 2017

One M/G/1/1 queue (1) and one M(s)/G/1 queue (2)

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

⇒ 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

One M/G/1/1 queue (1) and one 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 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 M/G/1/1 queue (1) and one 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.

⇒ The effective arrival process to queue 2 is not renewal.
Application: Hassin & S. 2017

One M/G/1/1 queue (1) and one 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.

⇒ 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 M/G/1/1 queue (1) and one 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.

⇒ 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

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.
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

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.
Summary of results

We introduce:
Summary of results

We introduce:

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