Dynamic Scheduling and Maintenance of a Deteriorating Server

Jefferson Huang

School of Operations Research & Information Engineering
Cornell University

June 19, 2018

INFORMS International Conference

Taipei, Taiwan

Joint work with Douglas Down (McMaster), Mark Lewis (Cornell), Cheng-Hung Wu (National Taiwan University)

Classic Problem: Scheduling Jobs

Classic Problem: Scheduling Jobs

Several types of jobs.

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see . . .

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see ...

Jobs of each type arrive, over time, at a single service station.

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see . . .

Jobs of each type arrive, over time, at a single service station.

- e.g., a machine toolset, a medical care provider, . . .

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see ...

Jobs of each type arrive, over time, at a single service station.

- e.g., a machine toolset, a medical care provider, ...

Arriving jobs can differ in:

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see ...

Jobs of each type arrive, over time, at a single service station.

- e.g., a machine toolset, a medical care provider, ...

Arriving jobs can differ in:

- service requirements

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see ...

Jobs of each type arrive, over time, at a single service station.

- e.g., a machine toolset, a medical care provider, ...

Arriving jobs can differ in:

- service requirements
- delay costs

Classic Problem: Scheduling Jobs

Several types of jobs.

- e.g., a manufacturer's work-in-process, patients to see ...

Jobs of each type arrive, over time, at a single service station.

- e.g., a machine toolset, a medical care provider, ...

Arriving jobs can differ in:

- service requirements
- delay costs

Problem: In what order should the jobs be worked on?

Classic Problem: Scheduling a Multiclass Queue

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs.

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

$$\text{Arrival times} \sim \text{point process on } \mathbb{R}_+ := [0, \infty).$$

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

$$\text{Arrival times} \sim \text{point process on } \mathbb{R}_+ := [0, \infty).$$

Arriving jobs need a random amount of service.

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

$$\text{Arrival times} \sim \text{point process on } \mathbb{R}_+ := [0, \infty).$$

Arriving jobs need a random amount of service.

$$\text{Service requirements} \overset{\text{iid}}{\sim} \text{exponential with mean 1.}$$

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

$$\text{Arrival times} \sim \text{point process on } \mathbb{R}_+ := [0, \infty).$$

Arriving jobs need a random amount of service.

$$\text{Service requirements} \overset{\text{iid}}{\sim} \text{exponential with mean 1.}$$

All jobs are processed by a single server.

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

$$\text{Arrival times} \sim \text{point process on } \mathbb{R}_+ := [0, \infty).$$

Arriving jobs need a random amount of service.

$$\text{Service requirements} \overset{\text{iid}}{\sim} \text{exponential with mean 1}.$$

All jobs are processed by a single server.

$$\text{Class } k \text{ jobs are served at rate } \mu_k.$$

Classic Problem: Scheduling a Multiclass Queue

K classes of jobs. In this talk, $K = 2$.

Jobs of each class arrive independently of the others.

Arrival times $\sim$ point process on $\mathbb{R}_+ := [0, \infty)$.

Arriving jobs need a random amount of service.

Service requirements $\overset{\text{iid}}{\sim}$ exponential with mean 1.

All jobs are processed by a single server.

Class k jobs are served at rate μ_k.

Waiting class k jobs incur holding costs at the (constant) rate c_k.

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$$\boxed{\text{expected long-run average cost per unit time.}}$$

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$\boxed{\text{expected long-run average cost per unit time.}}$

The following static priority policy is optimal (Nain 1989):

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$\boxed{\text{expected long-run average cost per unit time.}}$

The following static priority policy is optimal (Nain 1989):

If $c_1\mu_1 \geqslant c_2\mu_2$, prioritize class 1; otherwise, prioritize class 2.

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$\boxed{\text{expected long-run average cost per unit time.}}$

The following static priority policy is optimal (Nain 1989):

If $c_1\mu_1 \geqslant c_2\mu_2$, prioritize class 1; otherwise, prioritize class 2.

(the $c\mu$-**rule**)

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$$\boxed{\text{expected long-run average cost per unit time.}}$$

The following static priority policy is optimal (Nain 1989):

If $c_1\mu_1 \geqslant c_2\mu_2$, prioritize class 1; otherwise, prioritize class 2.

(the $c\mu$-**rule**)

Proof uses a change-of-measure result for Poisson processes to show that the original problem is equivalent to a reward-maximization problem.

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$\boxed{\text{expected long-run average cost per unit time.}}$

The following static priority policy is optimal (Nain 1989):

If $c_1\mu_1 \geqslant c_2\mu_2$, prioritize class 1; otherwise, prioritize class 2.

(the $c\mu$-**rule**)

Proof uses a change-of-measure result for Poisson processes to show that the original problem is equivalent to a reward-maximization problem.

- Reward rate of $c_k\mu_k$ when a class k job is being served.

Classic Problem: Scheduling a Multiclass Queue

Objective: Find a scheduling policy that minimizes the

$\boxed{\text{expected long-run average cost per unit time.}}$

The following static priority policy is optimal (Nain 1989):

If $c_1\mu_1 \geqslant c_2\mu_2$, prioritize class 1; otherwise, prioritize class 2.

(the $c\mu$-**rule**)

Proof uses a change-of-measure result for Poisson processes to show that the original problem is equivalent to a reward-maximization problem.

- Reward rate of $c_k\mu_k$ when a class k job is being served.
- Allows one to use an interchange argument on the sample paths of the process.

Twist: Time-Varying Service Rates

Twist: Time-Varying Service Rates

What if the service rates vary over time?

Twist: Time-Varying Service Rates

What if the service rates vary over time?

- e.g., the server's condition deteriorates

Twist: Time-Varying Service Rates

What if the service rates vary over time?

- e.g., the server's condition deteriorates

Assumptions:

Twist: Time-Varying Service Rates

What if the service rates vary over time?

- e.g., the server's condition deteriorates

Assumptions:

- the set $\mathcal{S}$ of possible server states is finite;

Twist: Time-Varying Service Rates

What if the service rates vary over time?

- e.g., the server's condition deteriorates

Assumptions:

- the set $\mathcal{S}$ of possible server states is finite;
- if the server state is $s \in \mathcal{S}$, it can serve class k jobs at rate μ_k^s;

Twist: Time-Varying Service Rates

What if the service rates vary over time?

- e.g., the server's condition deteriorates

Assumptions:

- the set $\mathcal{S}$ of possible server states is finite;
- if the server state is $s \in \mathcal{S}$, it can serve class k jobs at rate μ_k^s;
- the server state evolves independently according to a continuous-time Markov chain (CTMC).

Twist: Time-Varying Service Rates

What if the service rates vary over time?

- e.g., the server's condition deteriorates

Assumptions:

- the set $\mathcal{S}$ of possible server states is finite;
- if the server state is $s \in \mathcal{S}$, it can serve class k jobs at rate μ_k^s;
- the server state evolves independently according to a continuous-time Markov chain (CTMC).

Is the "$c\mu$-rule" optimal?

Scheduling with Time-Varying Service Rates

Scheduling with Time-Varying Service Rates

Here, the "$c\mu$-rule" means that if the server state is s,

Scheduling with Time-Varying Service Rates

Here, the "$c\mu$-rule" means that if the server state is s,

prioritize class 1 if $c_1 \mu_1^s \geqslant c_2 \mu_2^s$, and prioritize class 2 otherwise.

Scheduling with Time-Varying Service Rates

Here, the "$c\mu$-rule" means that if the server state is s,

prioritize class 1 if $c_1\mu_1^s \geqslant c_2\mu_2^s$, and prioritize class 2 otherwise.

Q: Does the "$c\mu$-rule" minimize the expected long-run average cost?

Scheduling with Time-Varying Service Rates

Here, the "$c\mu$-rule" means that if the server state is s,

prioritize class 1 if $c_1\mu_1^s \geqslant c_2\mu_2^s$, and prioritize class 2 otherwise.

Q: Does the "$c\mu$-rule" minimize the expected long-run average cost?

A: No!

Suboptimality of the $c\mu$-Rule

Suboptimality of the $c\mu$-Rule

Example:

Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.

Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.

Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.
  - Server State Process: CTMC with jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and equal holding time rates.

Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.
  - Server State Process: CTMC with jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and equal holding time rates.

The $c\mu$-rule (static priority to class 1) leads to an **unstable system**!

Suboptimality of the $c\mu$-Rule

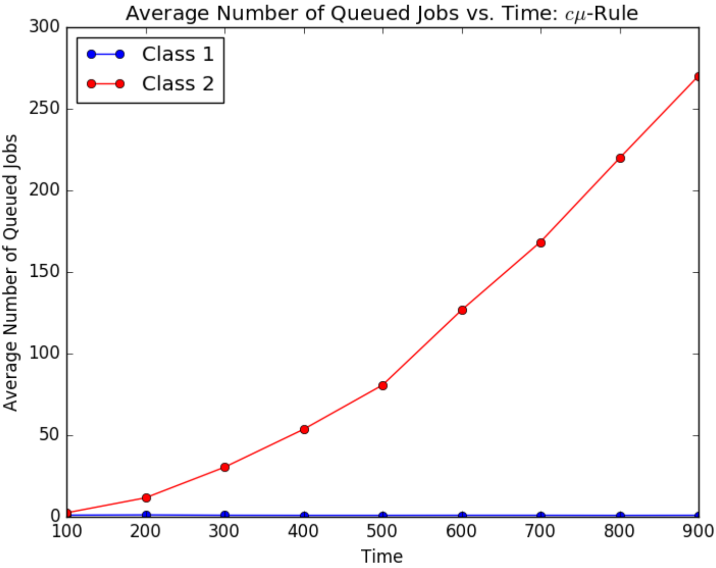
Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.
  - Server State Process: CTMC with jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and equal holding time rates.

The $c\mu$-rule (static priority to class 1) leads to an **unstable system**!

- has infinite expected long-run average cost!

Suboptimality of the $c\mu$-Rule



Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.
  - Server State Process: CTMC with jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and equal holding time rates.

The $c\mu$-rule (static priority to class 1) leads to an **unstable system**!

- has infinite expected long-run average cost!

At the same time, there is a policy that leads to a **stable system**!

Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.
  - Server State Process: CTMC with jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and equal holding time rates.

The $c\mu$-rule (static priority to class 1) leads to an **unstable system**!

- has infinite expected long-run average cost!

At the same time, there is a policy that leads to a **stable system**!

- e.g., if the server state is $s \in \{1, 2\}$, prioritize class s.

Suboptimality of the $c\mu$-Rule

Example:

- Arrivals: Independent Poisson processes with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- Server States: $\mathcal{S} = \{1, 2\}$
  - Service Rates: $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.
  - Server State Process: CTMC with jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and equal holding time rates.

The $c\mu$-rule (static priority to class 1) leads to an **unstable system**!

- has infinite expected long-run average cost!

At the same time, there is a policy that leads to a **stable system**!

- e.g., if the server state is $s \in \{1, 2\}$, prioritize class s.
- has finite expected average cost!

When is the $c\mu$-Rule Optimal?

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Theorem (H. et al. 2018)

If Assumption CR holds, then the $c\mu$-rule minimizes the expected average cost per unit time.

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Theorem (H. et al. 2018)

If Assumption CR holds, then the $c\mu$-rule minimizes the expected average cost per unit time.

Assumption CR ensures that an interchange argument can be used.

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Theorem (H. et al. 2018)

If Assumption CR holds, then the $c\mu$-rule minimizes the expected average cost per unit time.

Assumption CR ensures that an interchange argument can be used.

Questions:

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Theorem (H. et al. 2018)

If Assumption CR holds, then the $c\mu$-rule minimizes the expected average cost per unit time.

Assumption CR ensures that an interchange argument can be used.

Questions:

- Is Assumption CR necessary?

When is the $c\mu$-Rule Optimal?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Theorem (H. et al. 2018)

If Assumption CR holds, then the $c\mu$-rule minimizes the expected average cost per unit time.

Assumption CR ensures that an interchange argument can be used.

Questions:

- Is Assumption CR necessary?
- What about conditions on the server state process?

Controlling the Server State

Controlling the Server State

It can make sense to allow interventions that change the server state.

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Assumptions: Each intervention:

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Assumptions: Each intervention:

1. incurs a fixed cost K

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Assumptions: Each intervention:

1. incurs a fixed cost K
2. brings the server offline for a random amount of time

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Assumptions: Each intervention:

1. incurs a fixed cost K
2. brings the server offline for a random amount of time

Questions:

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Assumptions: Each intervention:

1. incurs a fixed cost K
2. brings the server offline for a random amount of time

Questions:

- When should an intervention be performed?

Controlling the Server State

It can make sense to allow interventions that change the server state.

- e.g., preventive maintenance of a deteriorating server

Assumptions: Each intervention:

1. incurs a fixed cost K
2. brings the server offline for a random amount of time

Questions:

- When should an intervention be performed?
- When it's not performed, which job class should be served?

Preventive Maintenance

Preventive Maintenance

Assumptions:

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,
$$0 < \mu_k^1 \leqslant \cdots \leqslant \mu_k^B < \infty.$$

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,
$$0 < \mu_k^1 \leqslant \cdots \leqslant \mu_k^B < \infty.$$

State $B =$ "like-new condition"

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,
$$0 < \mu_k^1 \leqslant \cdots \leqslant \mu_k^B < \infty.$$

State B = "like-new condition"

State 0 = "down for maintenance"

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,
$$0 < \mu_k^1 \leqslant \cdots \leqslant \mu_k^B < \infty.$$

State B = "like-new condition"

State 0 = "down for maintenance"

Transition to 0 without intervention = "failure"

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,
$$0 < \mu_k^1 \leqslant \cdots \leqslant \mu_k^B < \infty.$$

State B = "like-new condition"

State 0 = "down for maintenance"

Transition to 0 without intervention = "failure"

Intervention = "initiate preventive maintenance"

Preventive Maintenance

Assumptions:

- set of all server states is $\mathcal{S} = \{0, 1, \ldots, B\}$.
- $\mu_1^0 = \mu_2^0 = 0$.
- For $k = 1, 2$,
$$0 < \mu_k^1 \leqslant \cdots \leqslant \mu_k^B < \infty.$$

State B = "like-new condition"

State 0 = "down for maintenance"

Transition to 0 without intervention = "failure"

Intervention = "initiate preventive maintenance"

- The successive lengths of time that the server is down for maintenance are independent and identically distributed.

When is $c\mu$-Based Scheduling Sufficient?

When is $c\mu$-Based Scheduling Sufficient?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

When is $c\mu$-Based Scheduling Sufficient?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Assumption QO: The decision-maker does not use queue-length information (i.e., is "queue-oblivious") in making intervention decisions.

When is $c\mu$-Based Scheduling Sufficient?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Assumption QO: The decision-maker does not use queue-length information (i.e., is "queue-oblivious") in making intervention decisions.

- e.g., maintenance decisions are based on a fixed state threshold, are calendar-based, are job-based, . . .

When is $c\mu$-Based Scheduling Sufficient?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Assumption QO: The decision-maker does not use queue-length information (i.e., is "queue-oblivious") in making intervention decisions.

- e.g., maintenance decisions are based on a fixed state threshold, are calendar-based, are job-based, ...

Theorem (H. et al. 2018)

For the joint scheduling and preventive maintenance problem, suppose Assumptions CR and QO hold.

When is $c\mu$-Based Scheduling Sufficient?

Assumption CR: The *ratio* between the service rates stays *constant*:

$$\frac{\mu_1^i}{\mu_2^i} = \frac{\mu_1^j}{\mu_2^j} \qquad \text{for all server states } i, j.$$

Assumption QO: The decision-maker does not use queue-length information (i.e., is "queue-oblivious") in making intervention decisions.

- e.g., maintenance decisions are based on a fixed state threshold, are calendar-based, are job-based, ...

Theorem (H. et al. 2018)

For the joint scheduling and preventive maintenance problem, suppose Assumptions CR and QO hold. Then it is without loss of optimality to always schedule according to the $c\mu$-rule.

Structure of Optimal Maintenance Decisions?

Structure of Optimal Maintenance Decisions?

Question: Is there an optimal policy with "nice" properties?

- Reduce the number of policies that need to be considered.
- Make computing an optimal policy easier.

Assumption: The arrival processes are Poisson processes

Structure of Optimal Maintenance Decisions?

Question: Is there an optimal policy with "nice" properties?

- Reduce the number of policies that need to be considered.
- Make computing an optimal policy easier.

Assumption: The arrival processes are Poisson processes

In this case, the joint scheduling & preventive maintenance problem is a semi-Markov decision process (SMDP).

Structure of Optimal Maintenance Decisions?

Question: Is there an optimal policy with "nice" properties?

- Reduce the number of policies that need to be considered.
- Make computing an optimal policy easier.

Assumption: The arrival processes are Poisson processes

In this case, the joint scheduling & preventive maintenance problem is a semi-Markov decision process (SMDP).

Idea: Use the theory of SMDPs to study the structure of optimal policies.

Monotone Maintenance Decisions

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p + 1 \text{ (or } p - 1)}$$

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p + 1 \text{ (or } p - 1\text{)}}$$

Q: Is there an optimal policy that is monotone in the queue lengths?

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p + 1 \text{ (or } p - 1)}$$

Q: Is there an optimal policy that is monotone in the queue lengths?

A: **Not always!** (Kaufman & Lewis 2007).

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p+1 \text{ (or } p-1)}$$

Q: Is there an optimal policy that is monotone in the queue lengths?

A: Not always! (Kaufman & Lewis 2007).

- May want to:

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p + 1 \text{ (or } p - 1\text{)}}$$

Q: Is there an optimal policy that is monotone in the queue lengths?

A: Not always! (Kaufman & Lewis 2007).

- May want to:
  1. maintain when there are no jobs;

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p + 1 \text{ (or } p - 1)}$$

Q: Is there an optimal policy that is monotone in the queue lengths?

A: Not always! (Kaufman & Lewis 2007).

- May want to:
  1. maintain when there are no jobs;
  2. not maintain when there are few jobs;

Monotone Maintenance Decisions

A joint scheduling & preventive maintenance policy is monotone in the parameter P if

$$\boxed{\text{maintain when } P = p \implies \text{maintain when } P = p + 1 \text{ (or } p - 1)}$$

Q: Is there an optimal policy that is monotone in the queue lengths?

A: Not always! (Kaufman & Lewis 2007).

- May want to:
  1. maintain when there are no jobs;
  2. not maintain when there are few jobs;;
  3. maintain when there are many jobs.

Monotone Maintenance Decisions

Monotone Maintenance Decisions

Q: Is there an optimal policy that is monotone in the server state?

Monotone Maintenance Decisions

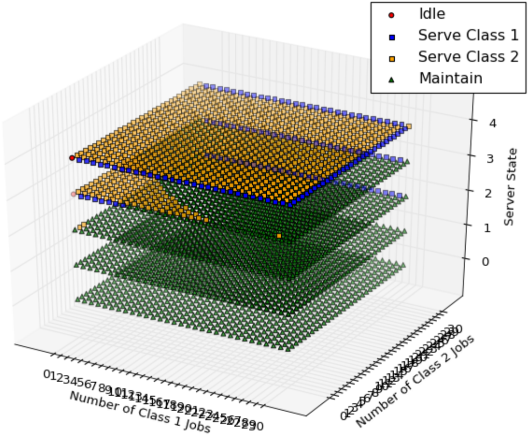
Q: Is there an optimal policy that is monotone in the server state?

A: **Yes!** (H. et al. 2018)

Monotone Maintenance Decisions

Q: Is there an optimal policy that is monotone in the server state?

A: Yes! (H. et al. 2018)



Conclusions & Possible Extensions

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.
   - Worth investing in making this the case?

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.
   - Worth investing in making this the case?
2. If (a) maintenance doesn't have visibility into the queue lengths, and (b) server state changes affect the server's capabilities uniformly, **stick with the $c\mu$-rule for the scheduling part**.

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.
   - Worth investing in making this the case?
2. If (a) maintenance doesn't have visibility into the queue lengths, and (b) server state changes affect the server's capabilities uniformly, **stick with the $c\mu$-rule for the scheduling part**.
3. Can work exclusively with policies that are **monotone in the server state**.

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.
   - Worth investing in making this the case?
2. If (a) maintenance doesn't have visibility into the queue lengths, and (b) server state changes affect the server's capabilities uniformly, **stick with the $c\mu$-rule for the scheduling part**.
3. Can work exclusively with policies that are **monotone in the server state**.

Some **possible extensions**:

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.
   - Worth investing in making this the case?
2. If (a) maintenance doesn't have visibility into the queue lengths, and (b) server state changes affect the server's capabilities uniformly, **stick with the $c\mu$-rule for the scheduling part**.
3. Can work exclusively with policies that are **monotone in the server state**.

Some **possible extensions**:

1. Class-dependent deterioration.

Conclusions & Possible Extensions

Some **practical takeaways** (pending more extensive empirical analysis)

1. If server state changes (a) cannot be controlled, and (b) affect the server's capabilities uniformly, **stick with the $c\mu$-rule**.
   - Worth investing in making this the case?
2. If (a) maintenance doesn't have visibility into the queue lengths, and (b) server state changes affect the server's capabilities uniformly, **stick with the $c\mu$-rule for the scheduling part**.
3. Can work exclusively with policies that are **monotone in the server state**.

Some **possible extensions**:

1. Class-dependent deterioration.
2. Partially observable server state.