Dynamic Scheduling and Maintenance of a Deteriorating Server

Jefferson Huang

School of Operations Research & Information Engineering
Cornell University

March 28, 2018

Seminar on Combinatorics, Games and Optimisation

London School of Economics and Political Science

A Single-Server Queue

The times between successive arrivals of jobs are random.

- T_i = time between $(i-1)^{\text{st}}$ and i^{th} arrival

The required service times of the arrivals are random.

- S_i = service time of the i^{th} arrival.

An "arrival" could be:

1. a customer/order;
2. someone calling a tech support hotline;
3. an injured person arriving at an emergency room; ...

A Single-Server Markovian Queue

For $i = 1, 2, \ldots,$

- T_i = time between $(i-1)^{\text{st}}$ and i^{th} arrival $\sim$ Exponential(λ)
- S_i = service time of the i^{th} arrival $\sim$ Exponential(μ)
- $T_1, T_2, \ldots$ and $S_1, S_2, \ldots$ are independent.

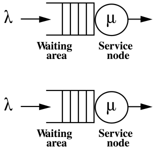
$Q(t)$ = number of jobs in the system at time $t \in [0, \infty) \in \{0, 1, 2, \ldots\}$

$$\{Q(t), t \geqslant 0\} \text{ is a continuous-time Markov chain (CTMC)}$$

- This kind of queue is called an "**M/M/1** queue".
- It's analytically tractable.

Parallel Markovian Queues

2 M/M/1 queues, 1 server:



The M/M/1 queues can have different parameters λ_1, λ_2 and μ_1, μ_2.

- Each queue holds a different class of arrival.

The server can only serve one arrival at a time.

Question: How should the server allocate its time?

Examples:

1. different kinds of orders;
2. different kinds of callers;
3. patients with ailments of varying severity, ...

Outline

Part 1: Optimally Scheduling Parallel Markovian Queues

Part 2: Optimal Scheduling with a Deteriorating Server

Part 3: Optimal Joint Scheduling and Maintenance

Part 1

Optimally Scheduling Parallel Markovian Queues

Costs

For each unit of time that an arrival in queue $i \in \{1, 2\}$ is in the system, a holding cost c_i is incurred.

A policy specifies, for all $t \in [0, \infty)$, which queue (if any) should be served at time t.

- Can be based on the past (but not future) evolution of the system.
- Should not idle the server when there are jobs waiting.

$Q_i^\pi(t)$ = number of class i jobs in the system at time t, under policy π

Objective: Find a policy π minimizing the long-run expected average cost

$$\limsup_{t\to\infty} \frac{1}{t}\mathbb{E}\left[\int_0^t \left(c_1 Q_1^\pi(x) + c_2 Q_2^\pi(x)\right)\ dx\right]$$

An Optimal Policy: The "$c\mu$-Rule"

c_i = holding cost rate for class i arrivals.

$1/\mu_i$ = expected service time for a class i arrival

$c\mu$-Rule: Prioritize class 1 if $c_1\mu_1 \geqslant c_2\mu_2$; otherwise, prioritize class 2.

Theorem (e.g., Buyukkoc Varaiya Walrand 1985)

The $c\mu$-rule is optimal, i.e., minimizes the long-run expected average cost.

Intuition

$c_i \mu_i$ is the expected cost decrease per class i job served

$A_i(t)$ = total number of class i arrivals during $[0, t]$

Assume there are no class i jobs in the system at time 0.

$$\text{total class } i \text{ cost up to time } t = \mathbb{E}\left[\int_0^t c_i Q_i^{\pi}(x)\ dx\right]$$
$$= \mathbb{E}\left[\int_0^t c_i A_i(x)\ dx\right]$$
$$- \mathbb{E}\left[\int_0^t c_i \mu_i \mathbf{1}\{\text{class } i \text{ served at time } x\}\ dx\right]$$

An "Interchange" Argument

Idea: Switching to the $c\mu$-rule never increases the cost incurred.

Example: One class 1 arrival, one class 2 arrival

$s_i = 1/\mu_i =$ service time of class i arrival

$$v_{1\to 2} = \text{cost to serve class 1, then class 2} = c_1 s_1 + c_2(s_1 + s_2)$$

$$v_{2\to 1} = \text{cost to serve class 2, then class 1} = c_2 s_2 + c_1(s_2 + s_1)$$

Then

$$v_{1\to 2} \leqslant v_{2\to 1} \iff c_1\mu_1 \geqslant c_2\mu_2$$

Can be made to work for every sample path of the process (e.g., Nain 1989).

A Mathematical Programming Argument

Idea: The problem can be formulated as a nice mathematical program. (e.g., Coffman Mitrani 1980)

$$x_i^\pi = \frac{\limsup_{t\to\infty} \frac{1}{t}\mathbb{E}\left[\int_0^t Q_i^\pi(x)\ dx\right]}{\mu_i}$$

$$= \text{long-run expected average amount of work in the system under policy } \pi$$

$$X = \{(x_1^\pi, x_2^\pi) : \pi \text{ is a policy}\}$$

Mathematical Programming Formulation:

minimize	$c_1\mu_1 x_1 + c_2\mu_2 x_2$
subject to	$(x_1, x_2) \in X$

A Mathematical Programming Argument

$\rho_i = \lambda_i/\mu_i$ = average class i utilization

It turns out that X is a line segment:

$$X = \left\{ (x_1, x_2) : x_1 \geqslant \frac{\rho_1/\mu_1}{1-\rho_1},\ x_2 \geqslant \frac{\rho_2/\mu_2}{1-\rho_2},\ x_1 + x_2 = \frac{\rho_1/\mu_1 + \rho_2/\mu_2}{1-\rho_1-\rho_2} \right\}$$

The extreme points of X correspond to priority policies:

$$\left(\frac{\rho_1/\mu_1}{1-\rho_1},\ \frac{\rho_1/\mu_1 + \rho_2\mu_2}{1-\rho_1-\rho_2} - \frac{\rho_1/\mu_1}{1-\rho_1} \right) \leftrightarrow \text{prioritize class 1}$$

$$\left(\frac{\rho_1/\mu_1 + \rho_2/\mu_2}{1-\rho_1-\rho_2} - \frac{\rho_2/\mu_2}{1-\rho_2},\ \frac{\rho_2\mu_2}{1-\rho_2} \right) \leftrightarrow \text{prioritize class 2}$$

A *Linear* Programming Argument

The solution to the linear program

$$\begin{array}{rl} \text{minimize} & c_1\mu_1 x_1 + c_2\mu_2 x_2 \\ \text{subject to} & (x_1, x_2) \in X \end{array}$$

is

$$\left(\frac{\rho_1/\mu_1}{1-\rho_1},\ \frac{\rho_1/\mu_1 + \rho_2\mu_2}{1-\rho_1-\rho_2} - \frac{\rho_1/\mu_1}{1-\rho_1}\right) \leftrightarrow \text{prioritize class 1}$$

if $c_1\mu_1 \geqslant c_2\mu_2$, and is

$$\left(\frac{\rho_1/\mu_1 + \rho_2/\mu_2}{1-\rho_1-\rho_2} - \frac{\rho_2/\mu_2}{1-\rho_2},\ \frac{\rho_2\mu_2}{1-\rho_2}\right) \leftrightarrow \text{prioritize class 2}$$

otherwise.

Part 2

Optimal Scheduling with a Deteriorating Server

A Deteriorating Server

What if the service time distributions vary with time?

$$S_i = \text{service time for class } i \text{ arrival} \sim \text{Exponential}(\mu_i(t)), \quad t \in [0, \infty).$$

Can reflect changes in the condition of the server.

Examples:

1. Machine processing parts on a manufacturing line, that is subject to wear (e.g., in a semiconductor wafer fab)
2. Human subject to fatigue (e.g. customer service rep, nurse)

Question: Given the state of the server, which class (if any) should be served?

A Natural Extension of the $c\mu$-Rule

$S(t)$ = state of the server at time $t \in [0, \infty)$

- Assume $\{S(t), t \geqslant 0\}$ is a continuous-time Markov chain.

Assume that if the server is in state s, then the service time of class i arrivals is

$$S_i \sim \text{Exponential}(\mu_i^s)$$

State-Dependent $c\mu$-Rule:

If the server is currently in state s, prioritize class 1 if

$$c_1 \mu_1^s \geqslant c_2 \mu_2^s;$$

otherwise, prioritize class 2.

Is the state-dependent $c\mu$-rule optimal?

The State-Dependent $c\mu$-Rule Can Be *Very* Suboptimal!

Example:

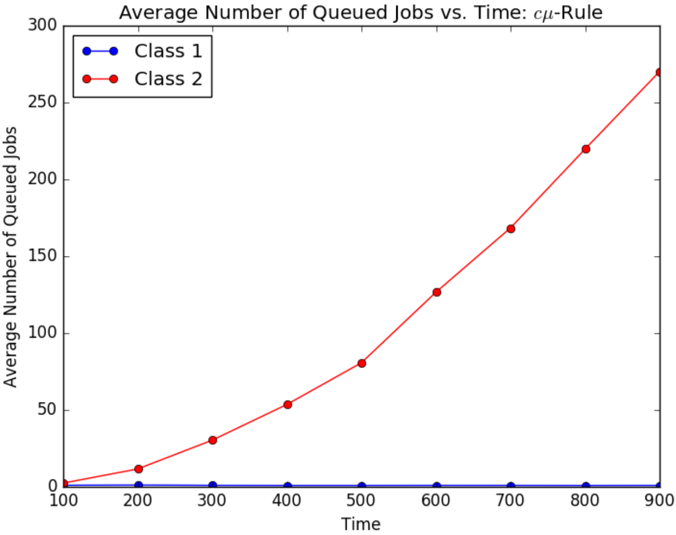
- Arrival Rates: $\lambda_1 = 5$, $\lambda_2 = 0.75$
- $\{S(t), t \geqslant 0\}$ cycles between states 1 and 2 at rate 1
- Service Rates:

$$\mu_1^1 = 10, \quad \mu_2^1 = 10$$
$$\mu_1^2 = 1, \quad \mu_2^2 = 2$$

Proposition (Huang et al. 2018)

1. *Under the state-dependent $c\mu$-rule, the long-run average number of jobs in queue 2 is infinite.*
2. *There exists a policy under which the long-run average numbers of jobs in both queues are finite.*

Unstable System Under the $c\mu$-Rule



Can the State-Dependent $c\mu$-Rule Be Optimal?

Constant-Ratio Assumption (CR): As the server changes states, the ratio between the service rates remains constant:

$$\frac{\mu_1^s}{\mu_2^s} = \frac{\mu_1^{s'}}{\mu_2^{s'}} \quad \forall \text{ server states } s, s'$$

(assume that $\mu_1^s = 0 \implies \mu_2^s = 0$ for all s)

Theorem (Huang et al. 2018)

If **(CR)** *holds, then the state-dependent $c\mu$-rule is optimal.*

- Doesn't depend on any parameters of the server state process.
- Under **(CR)**, a version of the classical interchange argument can be used.
- If **(CR)** is violated, then the state-dependent $c\mu$-rule may be very suboptimal (see preceding slide).

Part 3

Optimal Joint Scheduling and Maintenance

Server Deterioration and Failure

Suppose the set of possible server states is

$$\{0, 1, 2, \ldots, B\}$$

where

- higher state = higher service rates ($\mu_i^{s-1} \leqslant \mu_i^s \; \forall s \geqslant 1,\; i \in \{1, 2\}$)
- 0 = server has failed ($\mu_1^0 = \mu_2^0 = 0$)
- B = server is in perfect condition

The server deteriorates, and maintenance is initiated upon reaching state 0:

- current state is $s \geqslant 1 \implies$ next state is $s - 1$
- current state is $0 \implies$ next state is B

Preventive Maintenance

Can bring the server down for preventive maintenance before it fails on its own.

- Pay cost K each time this is done.

If there are jobs in the system, and the server has not failed on its own, the decision-maker can elect to either

1. serve one of the classes present, or
2. initiate preventive maintenance.

Question: How should the decision-maker jointly allocate the server's time and make maintenance decisions?

This is a difficult problem! (e.g., Kaufman Lewis 2007)

When Do Simple Scheduling Policies Suffice?

A maintenance policy stipulates whether preventive maintenance should be initiated.

A maintenance policy is queue-oblivious if it does not depend on the queue lengths.

- e.g., a threshold policy: initiate maintenance iff. the server state $s < s^*$

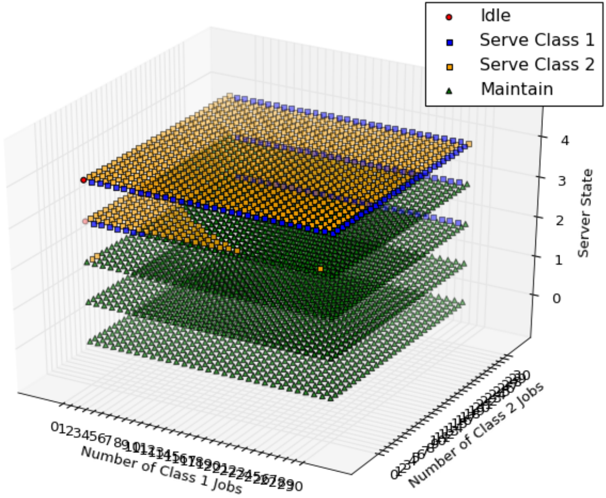
Theorem (Huang et al. 2018)

If one is restricted to queue-oblivious maintenance policies, then scheduling according to the state-dependent $c\mu$-rule is optimal.

- Show that under any fixed queue-oblivious maintenance policy, scheduling according to the state-dependent $c\mu$-rule is optimal.
- Classic interchange approach doesn't work if maintenance policies need not be queue-oblivious!

What's the Structure of Optimal Policies?

When is there an optimal policy with a nice structure?



Conclusions

Some conclusions on scheduling parallel Markovian queues:

1. When the server is reliable, the $c\mu$-rule is optimal.
2. When the server is unreliable, the state-dependent $c\mu$ rule can be very bad.
   - We provided a condition under which it's optimal.
3. The joint scheduling and maintenance problem is difficult.
   - We gave a partial result on the optimality of $c\mu$-based scheduling.

Research Directions:

1. Heuristics with performance guarantees
2. State-dependent deterioration
3. Non-exponential interarrival times and service times