Dynamic Scheduling and Maintenance of a Deteriorating Server

Jefferson Huang

School of Operations Research & Information Engineering
Cornell University

April 21, 2018

AMS Spring Eastern Sectional Meeting

Boston, MA

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

Scheduling Service in a Multiclass Queue

2 classes of jobs.

Jobs of each class arrive independently of the others.

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

Arriving jobs needs a random amount of service.

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

All jobs are processed by a single server.

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

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

Scheduling Service in a Multiclass Queue

A static priority policy minimizes the expected total cost incurred over any finite planning horizon (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.

This reformulation allows one to use an interchange argument on the sample paths of the process.

Time-Varying Service Rates

What if the service rates vary over time?

- deterioration of the server (e.g., testing unit in semiconductor manufacturing)

We **assume** that

- the server can be in one of a finite set of states;
- if the server state is s, its class k service rate is μ_k^s;
- the server state evolves according to a continuous-time Markov chain.

Could the $c\mu$-rule be optimal here?

Scheduling with Time-Varying Service Rates

Here, the "$c\mu$-rule" means:

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.

Question: Is this policy be optimal?

Answer: No, without additional assumptions!

Suboptimality of the $c\mu$-Rule

Example:

- Poisson arrivals to class k with rates $\lambda_1 = 5$, $\lambda_2 = 0.75$.
- 2 server states, jump matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, equal holding time rates.
- service rates $\mu_1^1 = \mu_1^2 = 10$, $\mu_2^1 = 1$, $\mu_2^2 = 2$.

The $c\mu$-rule (static priority to class 1) is unstable!

At the same time, a stable policy exists!

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

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 total cost incurred during any finite planning horizon.

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

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

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

Assume that each intervention

1. incurs a fixed cost K, and
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

Assume:

- Server states are numbered $0, 1, \ldots, B$.
- $\mu_1^0 = \mu_2^0 = 0$.
- 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 from server state 1 to 0 = "failure"

Intervention = "initiate preventive maintenance"

- The (random) times that the server is down for maintenance are iid.

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 for any finite planning horizon, it is without loss of optimality to always schedule according to the $c\mu$-rule.

Structure of Optimal Maintenance Decisions?

Assume:

- Poisson arrivals.
- Decisions are made whenever an event (arrival, service completion, server state change) occurs.
- Costs are continuously discounted over an infinite planning horizon.

The joint scheduling & preventive maintenance problem is a discounted semi-Markov decision process (SMDP).

Question: Is there an optimal policy with “nice” properties?

Monotone Maintenance Decisions

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

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

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

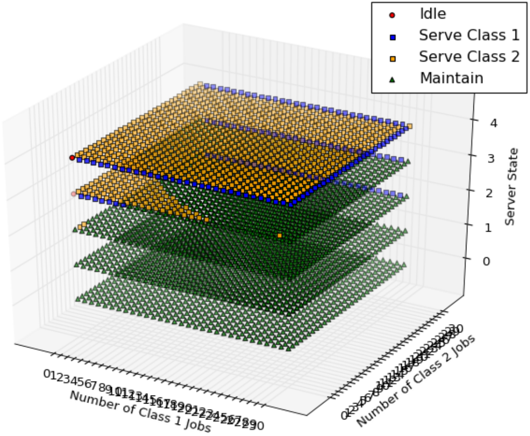
Answer: Not necessarily. (Kaufman & Lewis 2007).

- May want maintain when there are no jobs, not maintain when there are few jobs, and maintain when there are many jobs.

Monotone Maintenance Decisions

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

Answer: **Yes** (H. et al. 2018), via a dynamic programming proof.



Conclusions

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

1. If (a) server state changes 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. Look for policies that are monotone in the server state.

Some **possible extensions**:

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