Are highly scalable sequential dispatching policies asymptotically optimal?

Esa Hyytiä[1] Rhonda Righter[2]

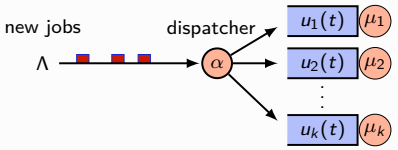
[1]University of Iceland / Aalto University
[2]University of California Berkeley

INFORMS APS 2023: June 29, Nancy, France

Outline:

Introduction

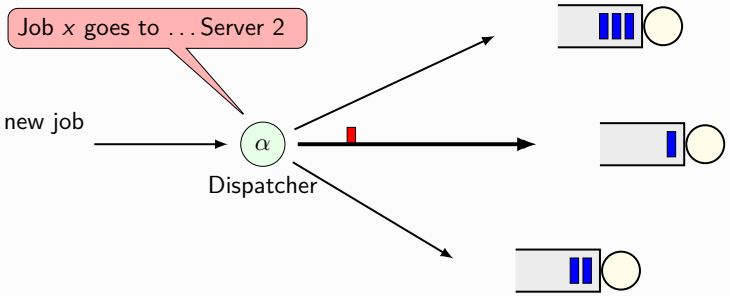
Dispatching Problem:



Families of Dispatching Policies:

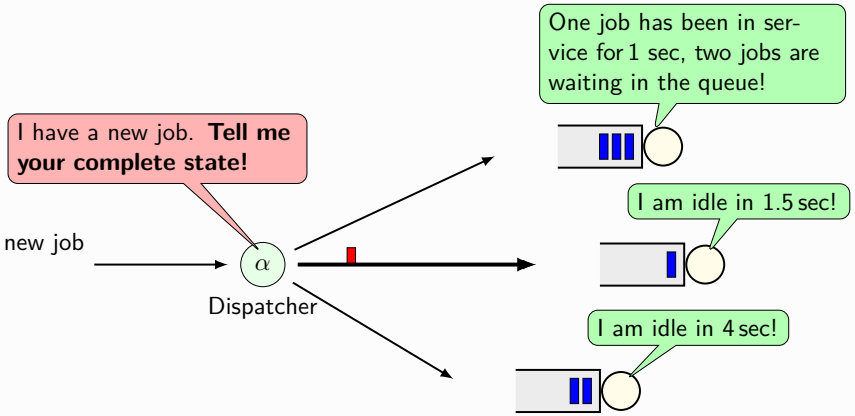
1. Static policies
   - No state information – scalable and allows multiple dispatchers
2. Dynamic policies
   - Join-the-shortest-queue (JSQ)
   - Least-work-left (LWL) (a.k.a. JSW)

Static Policies



"Fast local decision at the dispatcher"

Dynamic Policies

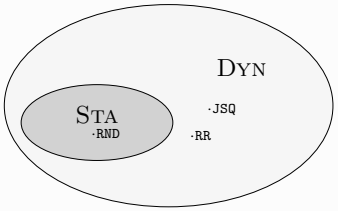


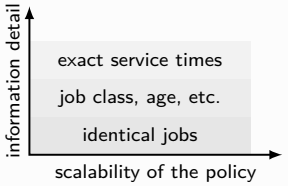
"Complete state-information can be retrieved and utilized"

Static and Dynamic Policies

Two basic classes:

- Static policies
- Dynamic policies





Optimal Dynamic Policy

- **Size-aware setting:**
  1. Size of the new job
  2. Backlogs in the servers
- **System state** $(\mathbb{R}^+)^k$
  - With the new job, $(\mathbb{R}^+)^{k+1}$ (decision point)
- **Optimality equations**
  - Value function $v(\mathbf{z}) : (\mathbb{R}^+)^k \to \mathbb{R}^+$ characterizes the optimal policy
  - Optimality equations can be derived (MDP formulation)
  - No closed-form solution is available
  - Numerical solutions can be computed (even though computationally heavy)

Analytical Solution (1)

Proposition (Optimality equation with Poisson arrivals)

The value function $v(\mathbf{z})$ of the optimal policy satisfies

$$\nabla_e v(\mathbf{z}) = \lambda\,(w^*(\mathbf{z}) - v(\mathbf{z}) - \mathbb{E}[W]) \tag{1}$$

where

$$w^*(\mathbf{z}) := \mathbb{E}[\min_i\{u_i + v(\mathbf{z} + X\mathbf{e}_i)\}] \tag{2}$$

with boundary conditions

$$\frac{\partial v(\mathbf{z})}{\partial u_i} = 0 \quad \textit{when } u_i = 0. \tag{3}$$

- state $\mathbf{z} = (u_1, \ldots, u_k)$ and $\mathbf{e}_i$ is the unit vector to direction i
- Poisson arrival process with rate λ (e.g. job/min),
- $\mathbb{E}[W]$ is the mean waiting time *(the performance metric)*
- X job size *(random variable with known distribution)*

Value iteration (with general i.i.d. IATs)

Let

- $a(t)$ is the pdf for the interarrival times A (and $\lambda = 1/\mathbb{E}[A]$)
- $f(x)$ is the pdf of the job sizes X
- $\mathbf{e} = (1, \ldots, 1)$, all-one vector

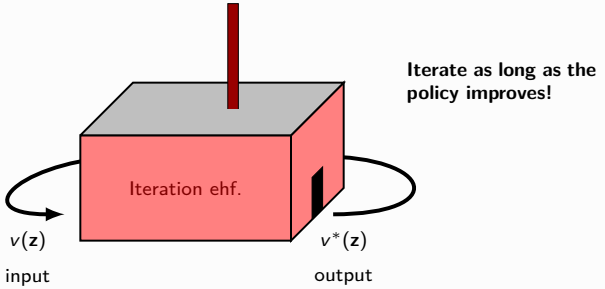
The value function can be obtained via value iteration:

$$\boxed{v(\mathbf{z}) \leftarrow \int_0^\infty a(t) w(\mathbf{z} - t\mathbf{e})\, dt} \tag{4}$$

where

$$w(\mathbf{z}) = \int_0^\infty f(x) \min_i \{u_i + v(\mathbf{z} + \mathbf{e}_i x)\}\, dx - w_0,$$
$$w_0 = \int_0^\infty f(x) v(\mathbf{e}_1 x)\, dx. \qquad (= \mathbb{E}[W]) \tag{5}$$

Value iteration in high level

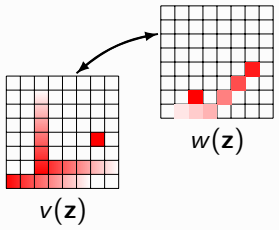


Iterate as long as the policy improves!

Input: *a function* $v(\mathbf{z})$ $(\mathbb{R}^k \to \mathbb{R})$
Output: *a better function* $v^*(\mathbf{z})$

Numerical solution

Integrals can be computed numerically:

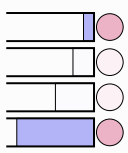


$$w_0 = \int_0^\infty f(x) v(\mathbf{e}_1 x)\, dx$$
$$w(\mathbf{z}) = \int_0^\infty \min_i f(x)[u_i + v(\mathbf{z} + \mathbf{e}_i x)] dx - w_0$$
$$v(\mathbf{z}) = \int_0^\infty a(t) w(\mathbf{z} - t\mathbf{e})\, dt$$

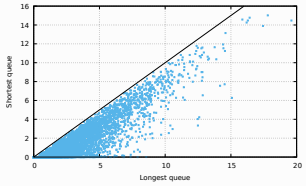
All this boils down to (many) **weighted sums over grid points!**

Example: 4 servers

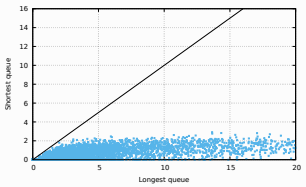
1. Shortest queue vs. the longest



LWL:



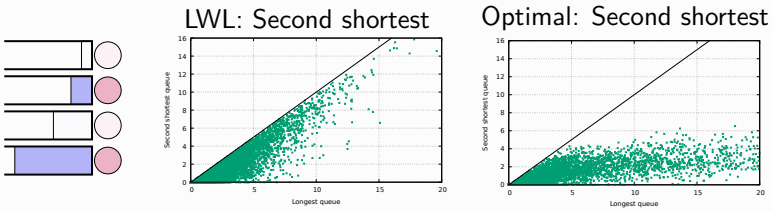
Optimal:



"Optimal policy unbalances the backlogs in the servers"

Example: 4 servers (2)

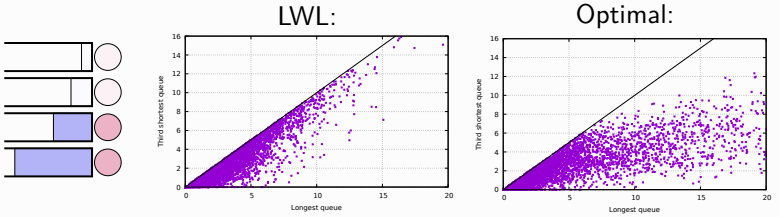
2. The second shortest queue vs. the longest



"Optimal policy unbalances the backlogs in the servers"

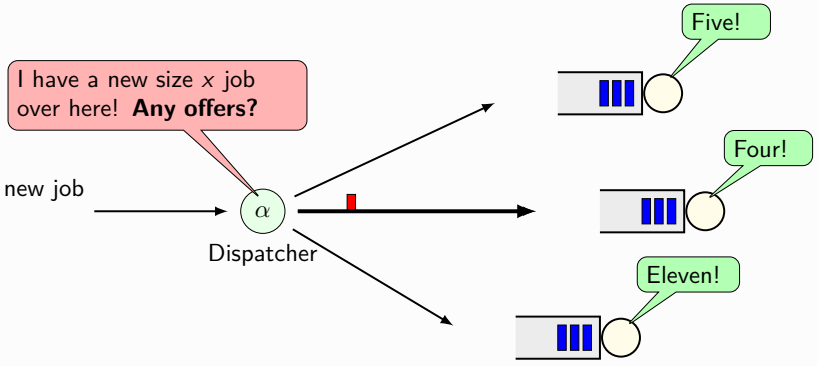
Example: 4 servers (3)

3. The third shortest queue vs. the longest

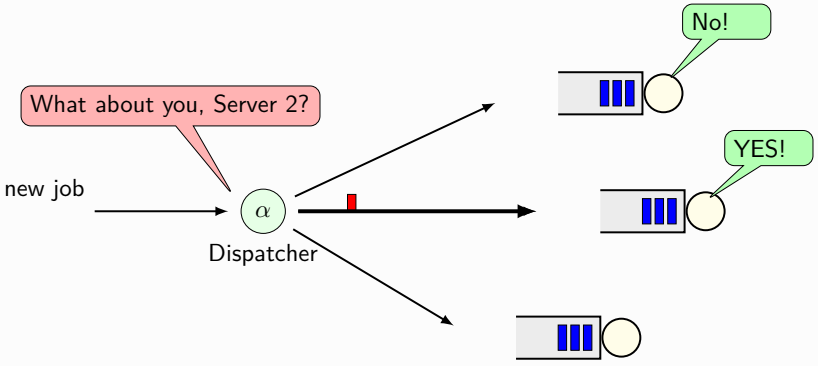


"Optimal policy unbalances the backlogs in the servers"

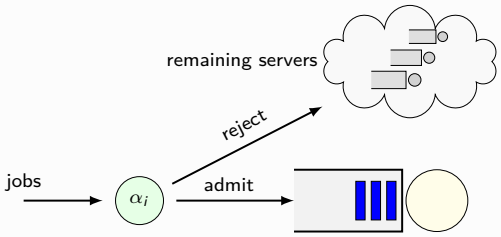
Index Policy



Sequential Policy

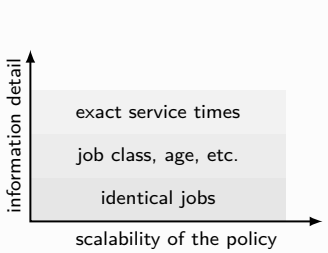


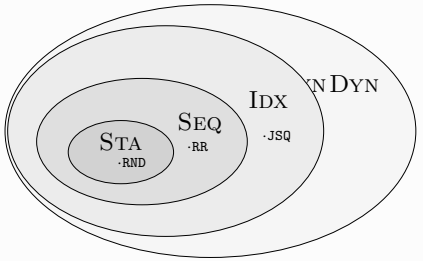
Sequential Policy



"Sequential policies consist of a cascaded set of admission policies"

Classification of Dispatching Policies





Theorem

STA $\subset$ SEQ $\subset$ IDX $\subset$ DYN.

Proof in the paper.

Fundamental question:

What is the optimal policy for each class?

Some results are available for certain classes/cases:

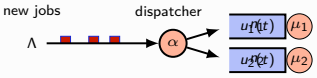
- STA: Size-Interval-Task-Assignment (SITA) (Feng et al. 2005)
- IDX: Join-the-shortest-queue (JSQ) for the Exp-case (Winston -77)
- DYN: Size-aware case via value iteration (numerically)

What about sequential policies?

Derivation of a sequential policy

Suppose

- We have two servers
- Poisson arrival process
- Static (basic) policy



In this case:

- Server 1 knows:
  1. its own backlog, and
  2. the distribution of the backlog in Server 2
- One policy iteration step can be carried out! (details in the paper)

Admission costs (from Server 1 point of view) are:

$$c_1 = u_1 + \frac{\lambda_1(2u_1x + x^2)}{2(1-\rho_1)}, \qquad c_2 = \mathbb{E}[W_2] + \frac{\lambda_2(2\,\mathbb{E}[W_2]x + x^2)}{2(1-\rho_2)}, \tag{6}$$

Heuristic Sequential Policy: **Server 1 accepts the job if $c_1 < c_2$.**

Numerical example: α_0 = SITA

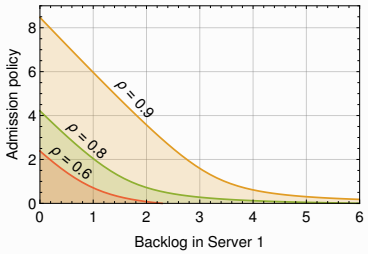
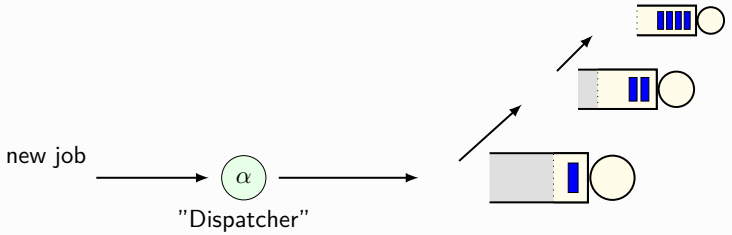


Figure: Sequential dispatching policies $\gamma_1(x)$ (admission to server 1) based on the first policy iteration.

- In all cases, the admission curve is **approximately a "triangle"**
- As if server 1 had a finite buffer!

Idea: Define a sequential policy by **finite buffers of increasing length!**

Family of ICE policies



Our new policies:

- **Slice**, the sequential ICE policy
- **Dice**, the dynamic ICE policy
  - Slice that relabels the servers in the increasing order of the backlog
- **Mice**, variant that *learns the virtual buffer lengths*

Simulation experiment: Two servers

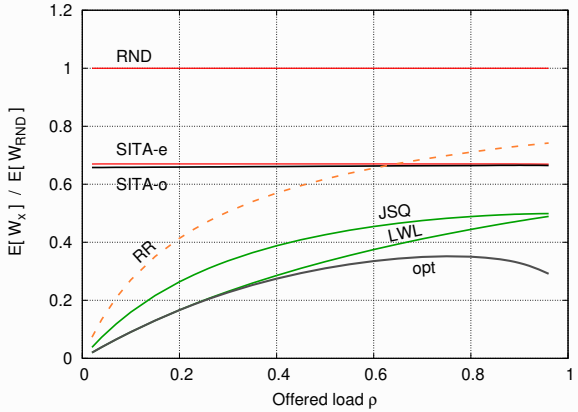


Figure: Simulation results with well-known reference policies.

Simulation experiment: Two servers (2)

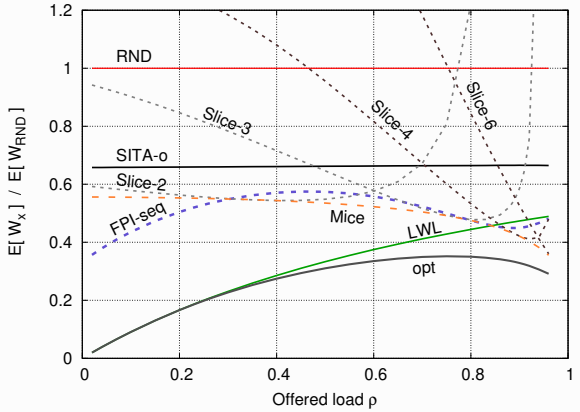


Figure: Simulation results with sequential policies.

Simulation experiment: Two servers (3)

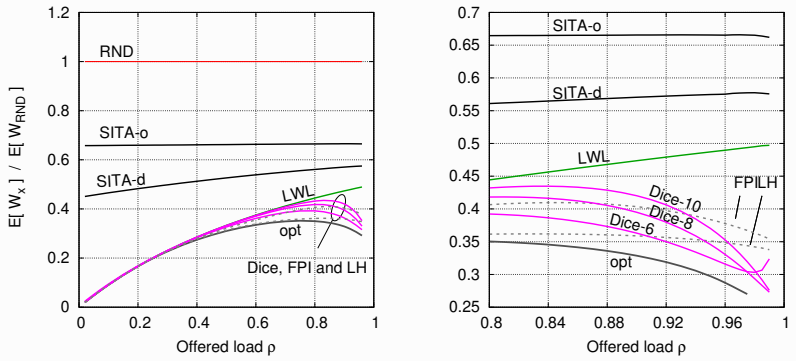
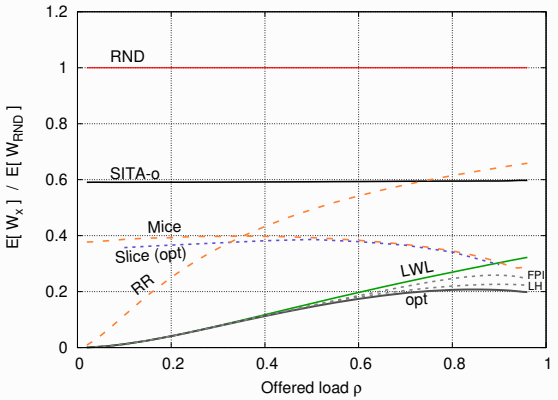


Figure: Simulation results for two servers with dynamic policies.

Simulation experiment: Three servers



Observation:

- Under heavy load, Slice and Mice again become near optimal!

Summary

1. **Classification of dispatching policies:**
   - Static, Sequential, Index and Dynamic classes

     $$\textsc{Sta} \subset \textsc{Seq} \subset \textsc{Idx} \subset \textsc{Dyn}.$$

   - Related to *scalability*
2. **Optimal policy** unbalances backlogs aggressively
3. **Sequential ICE-policies**:
   - One parameter for each server (block) to tune (cf. machine learning)
   - Surprisingly good performance in heavy traffic regime
   - With relabeling, excellent performance across all load levels
   - Unbalancing the backlogs by "finite virtual buffers" seems sufficient!
4. **Open Question:**
   As $\rho \to 1$, is the (optimal) sequential policy as good as the optimal dynamic policy?

Thanks!