Beyond Shortest Queue Routing

Esa Hyytiä, Rhonda Righter, Sigurður Gauti Samúelsson
University of Iceland
U.C. Berkeley
Aalto University

Outline

Problem introduction:

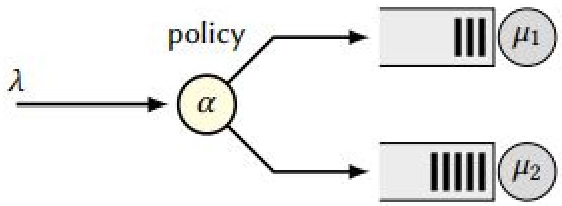
Modelling approach:

Policy iteration & general cost functions:

Numerical examples:

Conclusions:

Introduction



Routing jobs to parallel servers is a common and important task in today's computer systems

Join-shortest-queue (JSQ) routing minimizes the mean response time when the servers are identical and service times are independent and exponentially distributed

Apart from that, only a few optimality results exist, due to complexities from the infinite state space

Our problem

We do not know the size of the jobs

Job inter-arrival times and service times are exponentially distributed

State information is the number of jobs at each server

For clarity, we consider two heterogeneous parallel servers subject to a large class of cost structures, e.g., by

$c(w)$ = cost when job waits time w

Modelling approach generalizes straightforwardly to $K > 2$ servers

Heterogeneous servers

JSQ has been shown to be optimal in some specific cases, but, this is no longer the case, e.g., when service rates are unequal

SED (Shortest expected delay) is also not optimal with heterogeneous servers, even with exponential assumptions

State of our system

The state of our system is (n,m), where n denotes the number of jobs in server 1, and m is the number of jobs in server 2

With exponential assumptions

State x = (n,m)

Fixed routing policy; standard markov process

Otherwise; standard MDP

Infinity is not good

Ok, so infinite state space is a problem…

… what can we do …. LET´S GET RID OF IT!

First idea: truncate the state space?

Introduces bias, especially when the load is moderate or high …

Modelling

Our approach:

1) Model the system accurately where decisions matter the most
    a) States with a small number of jobs
2) Rely on appropriate approximations elsewhere
    a) State-space aggregation

This enables us to

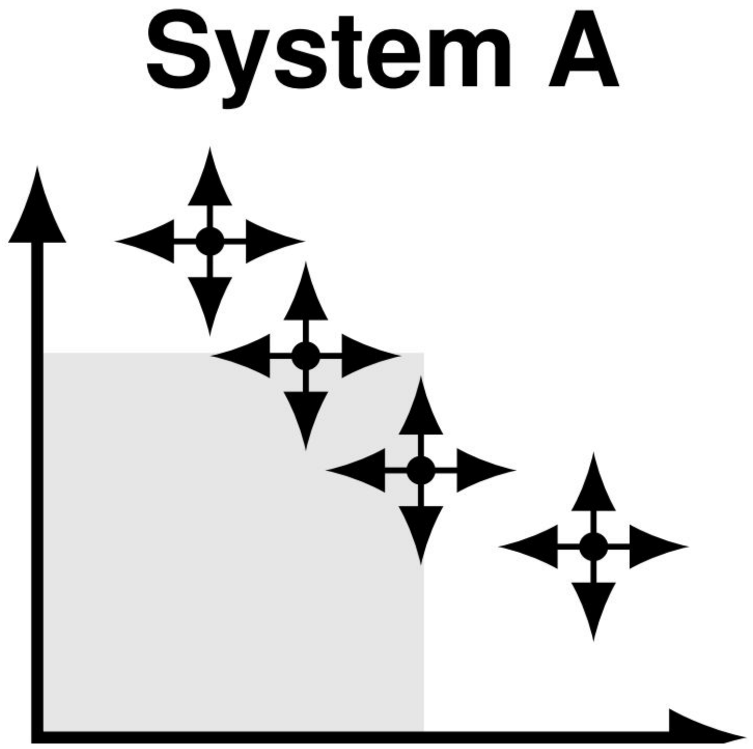
1) Analyze the system (with given routing policy) and
2) Compute the (near) optimal routing policy

Original system

Simple system, jobs arrive according to Poisson process and service time exponentially distributed

Servers are heterogeneous and we have a number-aware setting in where state n = (i,j) server 1 has i job’s, and server 2 has j job’s

Finding the optimal policy is surprisingly difficult due to the infinite state space



Fixed policy

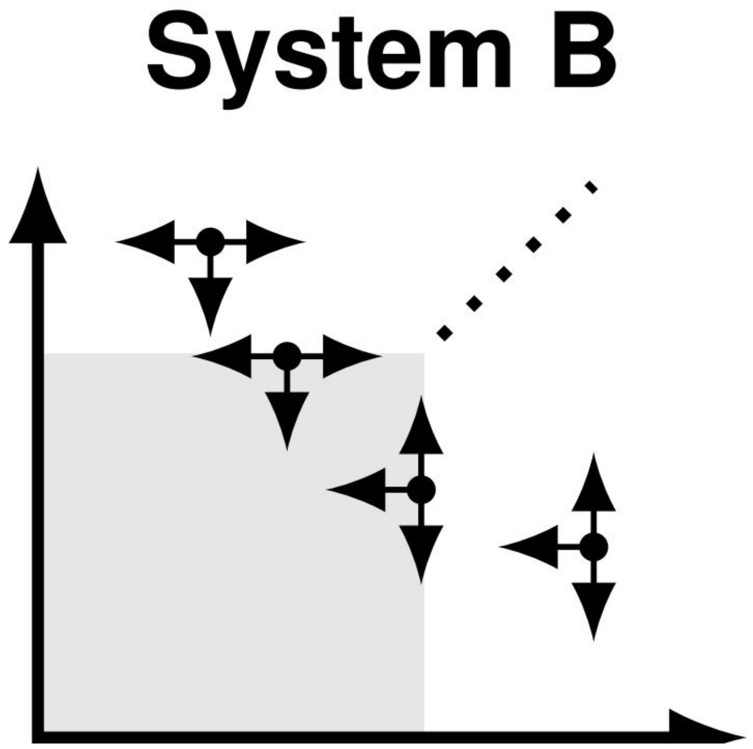
Now we modify the system

Routing is most crucial when servers have only a few jobs

Define a region A which contains a finite number of states near the origin

A = { (i, j) | i < n, j < m},
where n and m are free variables

Elsewhere, a default policy kicks in



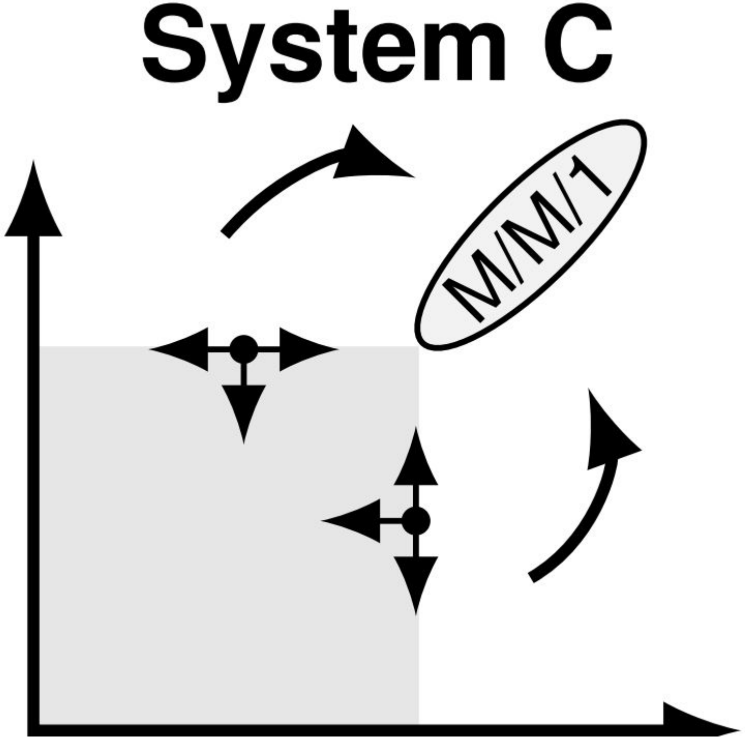
State space collapse/aggregation

When i,j >> 0, the system tends to stay near the diagonal line i ~ j

State-space collapse occurs beyond A

States beyond A “collapse” to M/M/1

- Service rate are combined
- Visit outside A = mini-busy period in M/M/1
- Equivalently, we allow jockeying



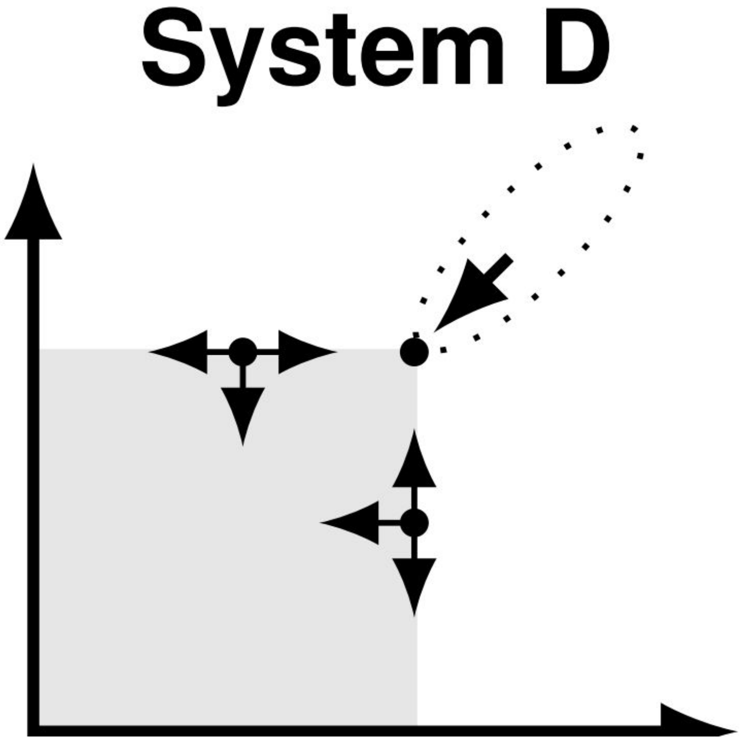
Combined super state

Mini-busy period of M/M/1 can be analyzed

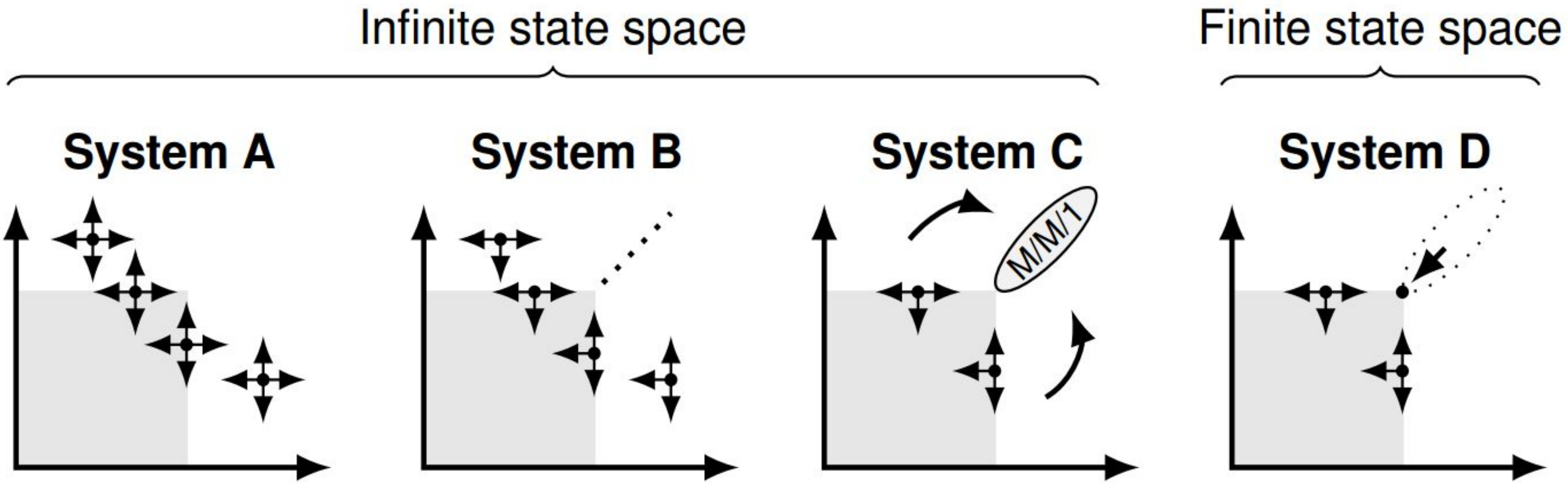
Aggregate M/M/1 to a new super state z

- With equal mean sojourn time
- With equal mean costs

Routing was fixed outside A, and thus Systems C and D are equivalent!



Our modelling systems



(a) Original system

(b) Fixed policy outside $\mathcal{A}$

(c) State-space collapse

(d) Combined super state

General cost function

Our approach allows for other performance metrics besides response time

For example, we may incur a unit cost if an arriving job sees more than two jobs ahead of itself upon arrival. This reflects how people tend to feel about queueing

General case with K servers

Fairly straightforward to extend to k > 2 servers

Problem is; by adding 1 server we are adding a whole new dimension to our problem

For small k, this approach seems feasible

For large k, *scalability* becomes an issue and the objective changes: routing is about finding an idle or sleeping server

This is a different problem

What can we do?

Can analyze any fixed policy

Develop better routing policies

Evaluating the approximation

Now we apply our approximation to estimate the mean response time with JSQ and SED

This validates the use of our system D and yields a sequence of increasingly more accurate estimates for the mean response time

Two identical servers with JSQ

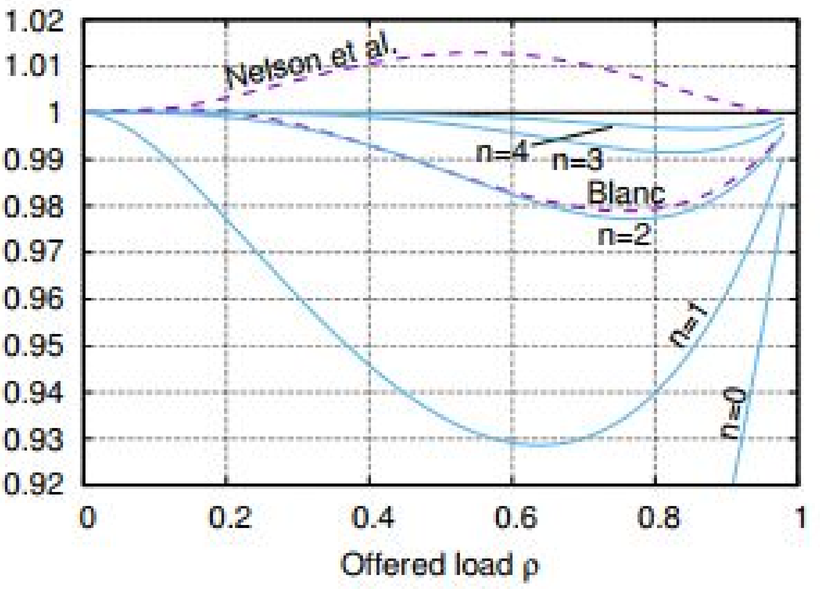
E[N] denotes the size of the square region A

$$\mathbb{E}[N_0] = \frac{\rho}{1-\rho} \qquad \text{(M/M/1)}$$

$$\mathbb{E}[N_1] = \frac{2\rho}{1-\rho^2} \qquad \text{(M/M/2)}$$

$$\mathbb{E}[N_2] = \frac{2\rho(2+(3-\rho)\rho(1+\rho))}{(1-\rho)(1+2\rho)(2+\rho+\rho^2)}$$

$$\mathbb{E}[N_3] = \frac{2\rho(4+\rho(14+\rho(23+\rho(16+7\rho-4\rho^3))))}{(1-\rho)(1+2\rho)(1+2\rho(1+\rho))(4+\rho(2+\rho+\rho^2))}$$



Near optimal routing

Sometimes it is convenient to assume that each customer incurs a cost upon arrival

For example, c(n) = (n+1)/(m mu) = mean response time of the new job

Equivalent cost rate r(n)=n

MDP with a finite state space

Value / policy iteration yields the optimal policy!

Numerical observations:

With example systems:

Numerical evidence on how quickly policy iteration converges

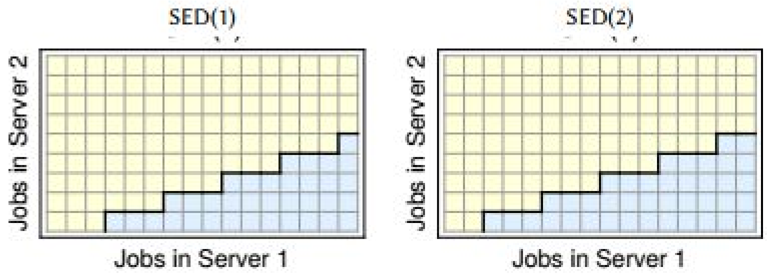
First policy iteration round yields the largest improvement

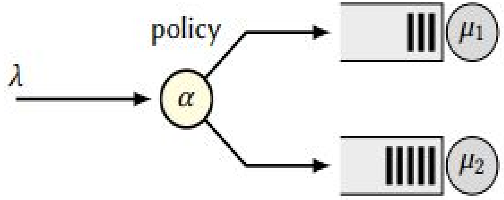
Numerical example:

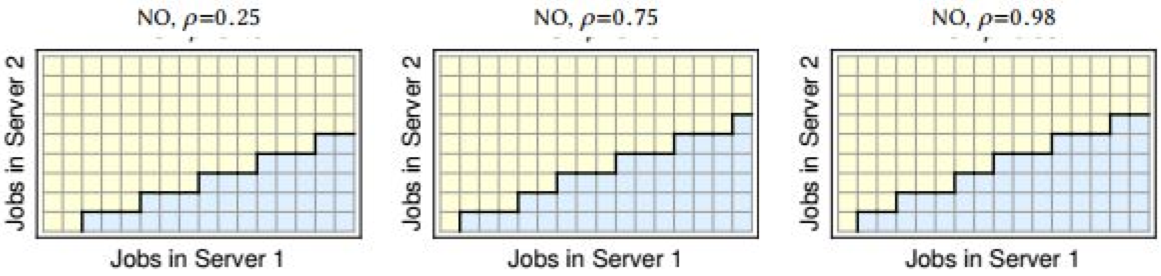
A closer look at SED and NO

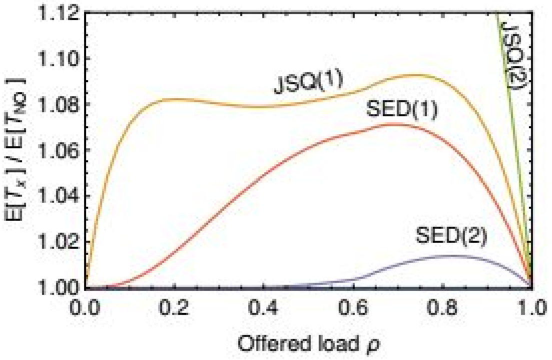
System:

1) Service rates (3,1)
2) Minimize mean response time









Conclusion

JSQ/SED are optimal only in specific cases

- Difficult problem with heterogeneous servers and arbitrary cost functions

New method developed that provides near-optimal routing policies

- Key idea: compress the infinite state space to a finite one

Our approach yields

- Closed form results and policies for small systems
- Easy and accurate numerical solutions for larger systems

Thank you for your attention.