Performance Evaluation 170 (2025) 102509

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Response time in a pair of processor sharing queues with Join-the-Shortest-Queue scheduling☆

Julianna Bor *, Peter G. Harrison

Department of Computing, Imperial College London, 180 Queen's Gate, London, SW7 2AZ, UK

ARTICLE INFO

Keywords:
Queueing theory
Response times
Generating functions

ABSTRACT

Join-the-Shortest-Queue (JSQ) is the scheduling policy of choice for many network providers, cloud servers, and traffic management systems, where individual queues are served under the processor sharing (PS) queueing discipline. A numerical solution for the response time distribution in two parallel PS queues with JSQ scheduling is derived for the first time. Using the generating function method, two partial differential equations (PDEs) are obtained corresponding to conditional response times, where the conditioning is on a particular traced task joining the first or the second queue. These PDEs are functional equations that contain partial generating functions and their partial derivatives, and therefore cannot be solved by commonly used techniques. We are able to solve these PDEs numerically with good accuracy and perform the deconditioning with respect to the queue-length probabilities by evaluating a certain complex integral. Numerical results for the density and the first four moments compare well against regenerative simulation.

1. Introduction

Join-the-Shortest-Queue (JSQ) is a scheduling policy that assigns arriving jobs to the shortest of a number of parallel queues and breaks ties by assigning jobs randomly with a given probability. JSQ – also called Least-Connection scheduling in a load balancing context – is the policy of choice for many network providers, cloud servers, and traffic management systems, where individual queues are served under processor sharing (PS) queueing discipline. For example, JSQ is offered by Cisco IOS Server [1], Azure Application Gateway [2], Kubernetes [3], and AWS Application Load Balancer [4]. Despite the ubiquity of JSQ scheduling, there is still a lot unknown regarding its behaviour quantitatively.

Consider a system of two parallel queues where Poisson arrivals occur at rate λ and join the shorter of the two queues, or, if the lengths are equal, join queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. Each queue has an independent server with exponential service times, and parameters μ_1, μ_2. In other words, consider a pair of parallel $M/M/1$ queues with JSQ scheduling. This apparently simple problem was first considered by Kendall almost 70 years ago in the symmetric case when the queueing discipline is FCFS [5], i.e. where $a_1 = a_2 = 1/2$ and $\mu_1 = \mu_2$. Later, Flatto and McKean obtained a complex closed-form solution for the queue-length distribution and mean response time in [6].

In a FCFS system, once a customer joins the queue, no later arriving customer can overtake it. This means that in order to calculate the cumulative distribution function of the time it takes the customer to leave the queue, there is no need to track the customers joining the queue after said customer. Therefore the calculation of response time distribution in an FCFS system is straightforwardly

☆ This article is part of a Special issue entitled: 'MASCOTS 2024' published in Performance Evaluation.

* Corresponding author.
E-mail addresses: jb1316@ic.ac.uk (J. Bor), pgh@ic.ac.uk (P.G. Harrison).

https://doi.org/10.1016/j.peva.2025.102509

Available online 2 September 2025

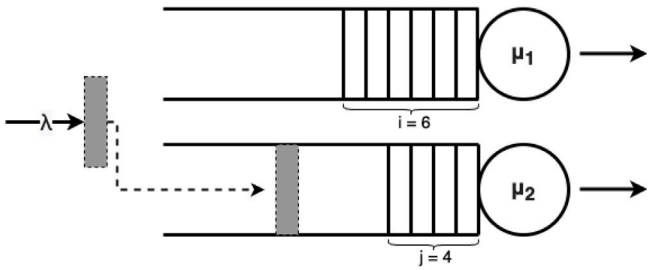

Fig. 1. Two JSQ-PS queues in parallel where the arriving task joins the shorter queue. The arrival rate is λ, the service rates are μ_1 and μ_2.

reduced to obtaining just the equilibrium queue length probabilities. Functional equations leading to the generating function of the equilibrium queue length probabilities for a pair of JSQ-FCFS queues have been obtained and solved recently in [7].

Now, take Kendall's model and modify it ever so slightly by replacing the FCFS queueing discipline with PS at the individual queues. This is a natural next step, as PS is often the preferred queueing discipline in practice. PS and FCFS have the same mean response time but the higher moments of PS are larger so that users experience greater variability. However, PS favours short jobs considerably and a user's mean response time at equilibrium is proportional to its own service requirement [8]. This inherent fairness is the reason why PS tends to be favoured over FCFS. Obtaining the response time distribution of the PS version of Kendall's 'simple' problem above remained unsolved for decades and is the focus of this paper. In fact, we solve a more general version of Kendall's model with PS queueing discipline.

The rest of the paper is organized as follows. Functional equations for the Laplace-Stieltjes transform (LST) of the response time distribution are obtained in Section 2. The functional equations are transformed into a problem in linear algebra and subsequently solved in Section 3. The model is evaluated in Section 4. Two optimization methods are introduced and subsequently evaluated in Section 5. Conclusions are drawn in Section 6.

2. Functional equations for the response time of two JSQ-PS queues

As outlined above, the queueing model under consideration consists of two parallel queues each with exponential service times and service rates of μ_1 and μ_2, respectively. Both queues use PS discipline locally. Arrivals are Poisson with rate λ, assuming $\lambda < \mu_1 + \mu_2$ for stability, and join the shortest queue. If at an arrival instant, the queues are of equal length, the arriving job is routed to queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. Fig. 1 displays the above network at an arrival instant when queue 1 has length $i = 6$ and queue 2 has length $j = 4$. The arriving task is therefore joining queue 2.

The goal is to derive the LST of the response time probability distribution, which is then inverted numerically to obtain the distribution (or density). In order to do that, we first derive two functional equations. These functional equations are then solved in Section 3.

2.1. Generating function for the queue-length distribution

To deal with response times at equilibrium, one first needs to obtain the joint steady-state queue-length distribution. This is derived for JSQ servers with FCFS queueing discipline in [7], but PS has exactly the same balance equations for the queue length probabilities in a Markovian model, and hence the same solution. Let $G(x, y)$ denote the generating function of the steady state queue-length probabilities, that is $G(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \pi_{i,j} x^i y^j$ where $\pi_{i,j}$ is the probability of i and j customers in the first and second queue, respectively, at equilibrium. In this paper, we assume $G(x, y)$ is known.

2.2. Generating functions of conditional response time densities

Our next step is to obtain functional equations for the generating functions of the LST of response time distributions, conditioned on whether the customer is in the first or the second queue. Let the random variables $U_{i,j}$, for $i, j \geq 0$, denote the remaining time to completion of service of a particular customer in queue 1 when there are i and j other customers present in queues 1 and 2, respectively. Let $U_{i,j}$ have probability distribution function $U_{i,j}(t)$, with LST $U^*_{i,j}(s)$. We define $V_{i,j}$, $V_{i,j}(t)$ and $V^*_{i,j}(s)$ analogously for a tagged customer in queue 2. We denote the generating function of $U^*_{i,j}(s)$ by $E(x, y, s)$ and the generating function of $V^*_{i,j}(s)$ by $F(x, y, s)$.

In order to derive functional equations for $E(x, y, s)$ and $F(x, y, s)$, in Definition 1, we define certain related Taylor series, each holomorphic in unit disks centred at the origin. Some of them are termed "partial generating functions" because they are power series in two variables that are not summed over the full first quadrant $\{(i, j) \mid 0 \leq i < \infty, 0 \leq j < \infty\}$. We use the following notation for functions $f(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{i,j} x^i y^j$ and for binary operator $\sim$:

$$f^{\sim} = \sum_{i \sim j} a_{i,j} x^i y^j$$

Note that $f^{\sim}$, where $\sim$ stands for one of $<, >, =$, can be expressed using $f^{\leq}(x,y)$, $f^{\geq}(x,y)$ and $f(x,y)$. For example, $f^{=}(x,y) = f(x,y)^{\leq} + f(x,y)^{\geq} - f(x,y)$.

Definition 1. Generating functions corresponding to $U^*_{i,j}(s)$ and $V^*_{i,j}(s)$, and related analytic functions:

1. Generating functions
$$E(x,y,s) := \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} U^*_{i,j}(s)x^i y^j$$
and $F(x,y,s)$ is defined similarly for $V^*_{i,j}(s)$.
2. Partial generating functions
$$E^{\leq}(x,y,s) := \sum_{i=0}^{\infty}\sum_{j=i}^{\infty} U^*_{i,j}(s)x^i y^j$$
Analogous expressions are defined for $E^{\geq}$, and for $F^{\leq}$, $F^{\geq}$, replacing U by V on the right-hand side.
3. Related analytic functions
$$\alpha_E(x,y,s) := \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} U^*_{i+1,j}(s)x^i y^j \qquad \gamma_E(x,y,s) := \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} U^*_{i+2,j}(s)x^i y^j$$
$$\beta_E(x,y,s) := \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} U^*_{i,j+1}(s)x^i y^j \qquad \delta_E(x,y,s) := \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} U^*_{i,j+2}(s)x^i y^j$$
and $\alpha_F(x,y,s), \beta_F(x,y,s), \gamma_F(x,y,s), \delta_F(x,y,s)$ are defined by replacing U by V on the right hand sides.
4. Partial versions of related analytic functions
$$\alpha_E^{\leq}(x,y,s) = \sum_{i=0}^{\infty}\sum_{j=i}^{\infty} U^*_{i+1,j}(s)x^i y^j \qquad \alpha_E^{\geq}(x,y,s) = \sum_{i=0}^{\infty}\sum_{j=0}^{i} U^*_{i+1,j}(s)x^i y^j$$
Similarly for $\beta_E^{\leq}, \beta_E^{\geq}, \gamma_E^{\leq}, \gamma_E^{\geq}, \delta_E^{\leq}, \delta_E^{\geq}$ and for $\alpha_F, \beta_F, \gamma_F, \delta_F$ by replacing U by V.

The following lemma shows how the related analytic functions can be expressed in terms of the original generating functions E and F. It is easy to derive by appropriate change of variable over the summation domains.

Lemma 1.
$$\alpha_E(x,y,s) = \frac{1}{x}\left[E(x,y,s) - E(0,y,s)\right] \qquad \gamma_E(x,y,s) = \frac{1}{x^2}\left[E(x,y,s) - E(0,y,s) - x\frac{\partial E}{\partial x}(0,y,s)\right]$$
$$\beta_E(x,y,s) = \frac{1}{y}\left[E(x,y,s) - E(x,0,s)\right] \qquad \delta_E(x,y,s) = \frac{1}{y^2}\left[E(x,y,s) - E(x,0,s) - y\frac{\partial E}{\partial y}(x,0,s)\right]$$
Analogous expressions can be obtained for $\alpha_F, \beta_F, \gamma_F, \delta_F$, replacing E by F on the right-hand sides.

Partial generating functions can be computed as follows [7]:
$$f^{\leq}(x_0,y_0) = \frac{1}{2\pi i}\oint_{C_y}\frac{f(x_0y_0/y,y)}{y-y_0}dy$$
where $f(x,y)$ is a two-parameter function that is analytic for all x, y with $|x| \leq r_1, |y| \leq r_2$. Furthermore, $|x_0| < r_1, |y_0| < r_2$ and the contour C_y is the circle with radius r_2. Again, a similar expression can be derived for $f^{\geq}$.

We now have the following result for the generating functions E and F corresponding to the LST of conditional response time distributions for a task joining queue 1 and queue 2, respectively.

Proposition 1. *The generating functions $E(x,y,s)$ and $F(x,y,s)$ satisfy the following equations:*
$$\begin{aligned}&\left(s+\lambda+(1-x)\mu_1+(1-y)\mu_2\right)\left(E(x,y,s)+x\frac{\partial E}{\partial x}(x,y,s)\right) - \mu_2\left(E(x,0,s)+x\frac{\partial E}{\partial x}(x,0,s)\right) - \lambda\frac{\partial E^{<}}{\partial x}(x,y,s)\\ &- \lambda\left(\beta_E^{\geq}(x,y,s)+x\frac{\partial \beta_E^{\geq}}{\partial x}(x,y,s)\right) - \lambda a_1\frac{\partial E^{=}}{\partial x}(x,y,s) - \lambda a_2 y\left(\delta_E^{=}(x,y,s)+x\frac{\partial \delta_E^{=}}{\partial x}(x,y,s)\right) = \frac{\mu_1}{(1-x)(1-y)}\end{aligned} \tag{1}$$
and
$$\begin{aligned}&\left(s+\lambda+(1-x)\mu_1+(1-y)\mu_2\right)\left(F(x,y,s)+y\frac{\partial F}{\partial y}(x,y,s)\right) - \mu_1\left(F(0,y,s)+y\frac{\partial F}{\partial y}(0,y,s)\right) - \lambda\frac{\partial F^{>}}{\partial y}(x,y,s)\\ &- \lambda\left(\alpha_F^{\leq}(x,y,s)+y\frac{\partial \alpha_F^{\leq}}{\partial y}(x,y,s)\right) - \lambda a_2\frac{\partial F^{=}}{\partial y}(x,y,s) - \lambda a_1 x\left(\gamma_F^{=}(x,y,s)+y\frac{\partial \gamma_F^{=}}{\partial y}(x,y,s)\right) = \frac{\mu_2}{(1-x)(1-y)}\end{aligned} \tag{2}$$

Proof. Suppose that at time t, the tagged customer is in queue 1 with i other customers and there are j customers at queue 2. Then, in the infinitesimal interval $(t, t+h]$ there are a number of possible events, one of which occurs with non-zero probability to first order in h:

1. An arrival to queue 1, if $i+1<j$, with probability λh;
2. An arrival to queue 2, if $i+1>j$, with probability λh;
3. An arrival to queue 1, if $i+1=j$, with probability $a_1\lambda h$;
4. An arrival to queue 2, if $i+1=j$, with probability $a_2\lambda h$;
5. A departure from queue 2, if $j>0$, with probability $\mu_2 h$;
6. A departure from queue 1 of a customer other than the tagged customer, if $i>0$, with probability $\frac{i}{i+1}\mu_1 h$;
7. Completion of the tagged customer's sojourn with probability $\frac{1}{i+1}\mu_1 h$;
8. No event occurs with probability $1-(\lambda+\mu_1+\mu_2 I_{j>0})h$

We therefore have the following equation to first order in h for $i,j\geq 0$, with terms ordered according to the above list of events:

$$\begin{aligned}U_{i,j}(t+h) &= h\lambda I_{i+1<j}U_{i+1,j}(t)+h\lambda I_{i+1>j}U_{i,j+1}(t)+h\lambda a_1 I_{i+1=j}U_{i+1,j}(t)+h\lambda a_2 I_{i+1=j}U_{i,j+1}(t)+h\mu_2 I_{j>0}U_{i,j-1}(t)\\ &\quad + h\mu_1 I_{i>0}\frac{i}{i+1}U_{i-1,j}(t)+\frac{h\mu_1}{i+1}+(1-h(\lambda+\mu_1+\mu_2 I_{j>0}))U_{i,j}(t)\end{aligned}$$

where I is the indicator function, i.e. I_B is 1 if B is true and 0 otherwise. Rearranging, dividing by h and taking the limit $h\to 0$ then yields

$$\begin{aligned}\frac{dU_{i,j}}{dt}(t) &= \lambda I_{i+1<j}U_{i+1,j}(t)+\lambda I_{i+1>j}U_{i,j+1}(t)+\lambda a_1 I_{i+1=j}U_{i+1,j}(t)+\lambda a_2 I_{i+1=j}U_{i,j+1}(t)+\mu_2 I_{j>0}U_{i,j-1}(t)+\mu_1 I_{i>0}\frac{i}{i+1}U_{i-1,j}(t)\\ &\quad +\frac{\mu_1}{i+1}-(\lambda+\mu_1+\mu_2 I_{j>0})U_{i,j}(t)\end{aligned}$$

Taking the LST and multiplying throughout by $(i+1)$ we get

$$\begin{aligned}(s+\lambda+\mu_1+\mu_2)(i+1)U^*_{i,j}(s)-\mu_2 I_{j=0}(i+1)U^*_{i,0}(s) &= \lambda I_{i+1<j}(i+1)U^*_{i+1,j}(s)+\lambda I_{i+1>j}(i+1)U^*_{i,j+1}(s)+\lambda a_1 I_{i+1=j}(i+1)U^*_{i+1,j}(s)\\ &\quad +\lambda a_2 I_{i+1=j}(i+1)U^*_{i,j+1}(s)+\mu_2 I_{j>0}(i+1)U^*_{i,j-1}(s)+\mu_1 I_{i>0}iU^*_{i-1,j}(s)+\mu_1\end{aligned}$$

where $U^*_{i,j}(s)$ is the LST of the distribution function $U_{i,j}(t)$. Multiplying by $x^i y^j$ and summing over $0\leq i,j<\infty$, with appropriate changes to these summation variables, yields (dropping the argument s for brevity):

$$\begin{aligned}(s+\lambda+\mu_1+\mu_2)&\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}x^i y^j\right)-\mu_2\left(x\sum_{i=0}^{\infty}U^*_{i,0}ix^{i-1}+\sum_{i=0}^{\infty}U^*_{i,0}x^i\right)=\lambda\sum_{i=1}^{\infty}\sum_{j=i+1}^{\infty}U^*_{i,j}ix^{i-1}y^j\\ &+\lambda\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{i}U^*_{i,j+1}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{i}U^*_{i,j+1}x^i y^j\right)+\lambda a_1\sum_{i=1}^{\infty}U^*_{i,i}ix^{i-1}y^i+\lambda a_2 y\left(x\sum_{i=0}^{\infty}U^*_{i,i+2}ix^{i-1}y^i+\sum_{i=0}^{\infty}U^*_{i,i+2}x^i y^i\right)\\ &+\mu_2 y\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}x^i y^j\right)+\mu_1 x\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}x^i y^j\right)+\mu_1\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}x^i y^j\end{aligned}$$

Plugging in the definitions for $E(x,y)$, $E^{\sim}(x,y)$, $\beta^{\sim}_E(x,y)$ and $\delta^{\sim}_E(x,y)$ and moving everything but the last term to the left-hand side, we get Eq. (1) as required.

The proof of Eq. (2) follows the exact same steps; it is omitted for brevity.

Note that the above equations have at most one solution due to the uniqueness of the steady-state distribution in the underlying Markov process. So far, we have derived functional equations for E in Eq. (1) and F in Eq. (2). Our next task is to combine these generating functions in order to express the LST of unconditional response time distribution.

2.3. LST of the unconditional response time distribution

The functional Eqs. (1) and (2) are solved for generating functions E and F in Section 3. However, it is still not straightforward to express the LST of unconditional response time distributions in terms of the above generating functions. This is because in the case of PS discipline overtaking might occur. Overtaking means that a customer that joins a queue can finish its service before some or all the customers that were already in the queue at the time of its arrival. It has a crucial effect as the rate at which a customer is being served at either queue at any one time is dependent on the number of customers in its queue—given i customers in queue 1, say, the rate at which a single customer is served is μ_1/i. The value of i in turn is dependent on the length of the other queue since arriving jobs always join the shorter of the two queues. Calculation of response time distribution under PS is therefore significantly more complicated than for FCFS. PS discipline requires the tracking of later arrivals to either queue whereas FCFS does not. As a result, the conditional LSTs are not geometric expressions and so more work is needed to perform the deconditioning. The required Laplace transform is given by the following proposition.

Proposition 2. *The Laplace transform of response time density in a pair of M/M/1 queues with JSQ-PS scheduling, omitting s from the integrands for brevity, is*

$$W^*(s) = \frac{1}{(2\pi)^2}\left(\int_0^{2\pi}\int_0^{2\pi} E^<(r_1e^{\imath t_1}, r_1e^{\imath t_2})G(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi} F^>(r_1e^{\imath t_1}, r_1e^{\imath t_2})G(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 \right.$$
$$\left. + \int_0^{2\pi}\int_0^{2\pi} a_1E^=(r_1e^{\imath t_1}, r_1e^{\imath t_2})G(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi} a_2F^=(r_1e^{\imath t_1}, r_1e^{\imath t_2})G(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 \right) \quad (3)$$

where $r_1 \le 1, r_2 = \frac{1}{r_1} \le \frac{\mu_1+\mu_2}{\lambda}$ and $\imath$ is the imaginary unit.

Proof.

$$W^*(s) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\left(I_{i<j}U^*_{i,j}(s) + I_{i>j}V^*_{i,j}(s) + I_{i=j}a_1U^*_{i,j}(s) + I_{i=j}a_2V^*_{i,j}(s)\right)\pi_{i,j}$$
$$= \sum_{i<j}U^*_{i,j}(s)\pi_{i,j} + \sum_{i>j}V^*_{i,j}(s)\pi_{i,j} + a_1\sum_{i=j}^{\infty}U^*_{i,j}(s)\pi_{i,j} + a_2\sum_{i=j}^{\infty}V^*_{i,j}(s)\pi_{i,j}$$

Omitting s for brevity and using the notation "$\sim$" as before for a binary operator, we focus on the $U^*_{i,j}$ terms first. In order to be able to express $\sum_{i\sim j}U^*_{i,j}\pi_{i,j}$, we need to extract coefficients that satisfy $i = i'$ and $j = j'$ from the below sum

$$E^{\sim}(x,y)G(z,q) = \sum_{i\sim j}\sum_{i'=0}^{\infty}\sum_{j'=0}^{\infty}U^*_{i,j}\pi_{i',j'}x^iy^jz^{i'}q^{j'}$$

The trick is a carefully chosen complex integral that filters out unwanted terms from the product of the generating functions. This can be achieved by applying Lemma 1 in [9], which is stated in a more generic setting. Here we provide an alternative, shorter argument for our specific case. Given $r_1 \le 1, r_2 = 1/r_1 \le \frac{\mu_1+\mu_2}{\lambda}$, we have

$$\frac{1}{(2\pi)^2}\int_0^{2\pi}\int_0^{2\pi} E^{\sim}(r_1e^{\imath t_1}, r_1e^{\imath t_2})G(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2$$
$$= \frac{1}{(2\pi)^2}\int_0^{2\pi}\int_0^{2\pi}\sum_{i\sim j}\sum_{i'=0}^{\infty}\sum_{j'=0}^{\infty}U^*_{i,j}\pi_{i',j'}r_1^ir_2^{i'}\left(e^{\imath t_1}\right)^{i-i'}r_1^jr_2^{j'}\left(e^{\imath t_2}\right)^{j-j'}dt_1dt_2$$
$$= \sum_{i\sim j}\sum_{i'=0}^{\infty}\sum_{j'=0}^{\infty}U^*_{i,j}\pi_{i',j'}r_1^{i+j}r_2^{i'+j'}\left(\frac{1}{2\pi}\int_0^{2\pi}\left(e^{\imath t_1}\right)^{i-i'}dt_1\right)\left(\frac{1}{2\pi}\int_0^{2\pi}\left(e^{\imath t_2}\right)^{j-j'}dt_2\right)$$
$$= \sum_{i\sim j}U^*_{i,j}\pi_{i,j}(r_1r_2)^{i+j} = \sum_{i\sim j}U^*_{i,j}\pi_{i,j}$$

where for the penultimate equality, we used the fact that

$$\int_0^{2\pi}e^{\imath tk}dt = \begin{cases} 0 & \text{if } k \neq 0 \\ 2\pi & \text{if } k = 0 \end{cases}$$

The sums containing the $V^*_{i,j}$ terms are obtained similarly.

In this procedure, the choices of r_1 and r_2 are important. The conditional Laplace transforms' generating functions ($E(x,y,s)$ and $F(x,y,s)$) diverge at $r_1 = 1$ when $s = 0$, each then being equal to $1/((1-x)(1-y))$. Similarly, $G(x,y)$ diverges at its radius of convergence, which we took to be $(\mu_1+\mu_2)/\lambda$. We chose a value for r_2 halfway between 1 and G's radius of convergence, i.e. $r_2 = \frac{\lambda+\mu_1+\mu_2}{2\lambda}$. However, note that for heavily loaded systems, $\lambda \simeq \mu_1+\mu_2$ so that r_1 ends up near 1 and r_2 near G's radius of convergence. Hence, significant numerical errors can be expected at high loads.

3. Solving the functional equations

In Section 2, we stated three functional equations for the generating functions of conditional and unconditional response time distribution LSTs, namely Eqs. (1), (2) and (3). To solve these functional equations numerically, we transform them into a problem in linear algebra.

Consider a bivariate functional equation of the form:

$$a(x,y)f(x,y) = b(x,y) + c(x,y)f(x,0) + d(x,y)\Delta_yf(0,y) \quad (4)$$

where $a(x,y)$, $b(x,y)$, $c(x,y)$ and $d(x,y)$ are known functions of x and y, and Δ_y denotes partial differentiation with respect to (wrt) y. This is a complex equation that might be solvable by Boundary Value Problem (BVP) methods, but a numerical solution would appear problematic. Our numerical approximation method is the following.

Let C_x and C_y be contours in the complex planes of x and y. Typically, C_x and C_y are circles. The interior of C_x is then a disk D_x, and similarly for C_y, D_y. Now, the value of a function $f(x,y)$ at any point (x_0,y_0) inside the Cartesian product of the

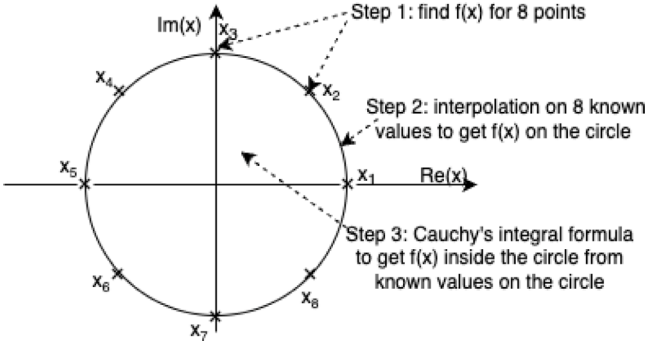

Fig. 2. Steps to approximate f(x) from 8 known values around the unit circle.

Table 1
Matrices representing operations in the summands of the functional equation for $f(x, y)$.

Operation	Matrix
Identity operator giving f	$\boldsymbol{I}^{nm\times nm}$
Multiplication by known function $A(x, y)$	$\boldsymbol{\Lambda}(A)$
Differentiation wrt x	$\boldsymbol{M}_{1,0}$
Differentiation wrt y	$\boldsymbol{M}_{0,1}$
Evaluation at an internal point $u_0 \in D_x$	$\boldsymbol{U}_{u_0}$
Evaluation at an internal point $v_0 \in D_y$	$\boldsymbol{V}_{v_0}$
Partialization giving $f^{\leq}$	$\boldsymbol{P}^{\leq}$
Partialization giving $f^{\geq}$	$\boldsymbol{P}^{\geq}$

contours, $C_x \times C_y$ (i.e. with $x_0 \in D_x$, and $y_0 \in D_y$), can be obtained by Cauchy's Integral Theorem applied twice in two dimensions, provided $f(x, y)$ is known for all $x \in C_x, y \in C_y$; see [10,11] for example. We therefore first seek an interpolation for f over $\{(x, y) \in C_x \times C_y\}$ by choosing two sets of discretization points $x_1, \ldots, x_m \in C_x$ and $y_1, \ldots, y_n \in C_y$. We take $m = n$ to simplify the notation; the analysis is no more complicated when $m \neq n$ but is more cluttered. Choosing C_x and C_y to be circles means the interpolation can be done easily on the real-valued arguments (angles) of the discretization points, giving more precision. Moreover, interpolation on reals is directly supported in most mathematical software packages. We found linear interpolation to be sufficient in our numerical experiments, producing very accurate results; however, higher-order methods could also be employed. An approximation to $f(x, b)$ on C_x, for fixed $b \in C_y$, is an interpolation over the points $x_1, \ldots, x_n$ of the values $f(x_1, b), \ldots, f(x_n, b)$, and similarly for $f(a, y)$ at fixed $a \in C_x$. We order the point-pairs alphabetically by subscript, $\{(x_1, y_1), \ldots, (x_1, y_n), (x_2, y_1), \ldots, (x_n, y_n)\}$ and say that $\vec{f} = (f(x_1, y_1), \ldots, f(x_1, y_n), f(x_2, y_1), \ldots, f(x_n, y_n))$ is a *vector representation* of f on $C_x \times C_y$. The vector representation can even be symbolic, i.e. $\vec{f} = (f_{11}, \ldots, f_{nn})$. Fig. 2 shows how the approximation works in one dimension. Note that, since we are initially interested in solving the functional equation for a finite set of input values, a linear approximation of all the operators present in the equation for $f(x, y)$ naturally leads to matrix representations. We refer to these as the operators' *matrix representations*, allowing us to rewrite the functional equations as a matrix–vector equation. For example, multiplication by a fixed function $a(x, y)$ is represented by a diagonal matrix of the multiplying function's values at the discretization point-pairs, ordered alphabetically, and we define

$$\boldsymbol{\Lambda}(\vec{a}) = \mathrm{Diag}\Big(a(x_i, y_j) \mid 1 \leq i \leq n, 1 \leq j \leq n\Big).$$

Matrices for the operators of differentiation and partialization (i.e. producing a partial generating function) were obtained in [7] by considering each point $x_1, \ldots, x_n, y_1, \ldots, y_n$ in turn, distorting the contour to go around it, and applying Cauchy's Integral Theorem. The details are not reproduced here, apart from Table 1, which lists the matrices corresponding to each operator of interest. Note that all the matrices in Table 1 are constant, applying to any functional equation solved for the given n discretization points in both complex planes. Eq. (4) above is thereby transformed to:

$$(\boldsymbol{\Lambda}(\vec{a}) - \boldsymbol{\Lambda}(\vec{c})\boldsymbol{V}_0 - \boldsymbol{\Lambda}(\vec{d})\boldsymbol{M}_{0,1}\boldsymbol{U}_0)\vec{f} = \boldsymbol{\Lambda}(\vec{b})$$

giving a vector representation

$$\big(\boldsymbol{\Lambda}(\vec{a}) - \boldsymbol{\Lambda}(\vec{c})\boldsymbol{V}_0 - \boldsymbol{\Lambda}(\vec{d})\boldsymbol{M}_{0,1}\boldsymbol{U}_0\big)^{-1}\boldsymbol{\Lambda}(\vec{b})$$

for the solution-function $f(x, y)$ on the contours.

Table 2
Matrices representing the operations corresponding to the related analytic functions of Definition 1.

Operation	Matrix
$\alpha_f(x,y) = \frac{1}{x}[f(x,y) - f(0,y)]$	$\boldsymbol{A}^{nm\times nm}$
$\beta_f(x,y) = \frac{1}{y}[f(x,y) - f(x,0)]$	$\boldsymbol{B}^{nm\times nm}$
$\gamma_f(x,y) = \frac{1}{x^2}\left[f(x,y) - f(0,y) - x\frac{\partial}{\partial x}f(0,y)\right]$	$\boldsymbol{C}^{nm\times nm}$
$\delta_f(x,y) = \frac{1}{y^2}\left[f(x,y) - f(x,0) - y\frac{\partial}{\partial y}f(x,0)\right]$	$\boldsymbol{D}^{nm\times nm}$

3.1. Conditional response times

In order to transform the equations for functions $E(x,y,s)$ and $F(x,y,s)$, we need to define new matrices for the related analytic functions given in Definition 1.

Proposition 3. *Define the following operators associated with function $f(x,y)$:*

$$\alpha_f(x,y) = \frac{1}{x}[f(x,y) - f(0,y)] \qquad \gamma_f(x,y) = \frac{1}{x^2}\left[f(x,y) - f(0,y) - x\frac{\partial}{\partial x}f(0,y)\right]$$
$$\beta_f(x,y) = \frac{1}{y}[f(x,y) - f(x,0)] \qquad \delta_f(x,y) = \frac{1}{y^2}\left[f(x,y) - f(x,0) - y\frac{\partial}{\partial y}f(x,0)\right]$$

and let $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ be the corresponding matrix representations, respectively, given by:

$$\boldsymbol{A} = \boldsymbol{\Lambda}(1/\vec{x})\left(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{U}_0\right) \qquad \boldsymbol{C} = \boldsymbol{\Lambda}(1/\vec{x}^2)(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{U}_0 - \boldsymbol{\Lambda}(\vec{x})\boldsymbol{U}_0\boldsymbol{M}_{1,0})$$
$$\boldsymbol{B} = \boldsymbol{\Lambda}(1/\vec{y})\left(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{V}_0\right) \qquad \boldsymbol{D} = \boldsymbol{\Lambda}(1/\vec{y}^2)(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{V}_0 - \boldsymbol{\Lambda}(\vec{y})\boldsymbol{V}_0\boldsymbol{M}_{0,1})$$

Equipped with these new matrices, the transformation of the equations for $E(x,y,s)$ and $F(x,y,s)$, albeit more complicated, follows the same procedure introduced above. Introducing the notation

$$\vec{c} = \left(\frac{1}{(1-x_i)(1-y_j)} \mid 1 \le i \le n, 1 \le j \le n\right) \tag{5}$$

where the column vector on the right-hand side is ordered alphabetically as usual, after replacing operators by their corresponding matrices in Tables 1 and 2, Eq. (1) becomes:

$$\boldsymbol{\Gamma}_E^{-1}(s)\vec{e} = \mu_1\vec{c} \tag{6}$$

where $\vec{e}$ is the vector representation of $E(x,y,s)$ and

$$\begin{aligned}\boldsymbol{\Gamma}_E^{-1}(s) =& \boldsymbol{\Lambda}(s+\lambda+(1-\vec{x})\mu_1+(1-\vec{y})\mu_2)\left(\boldsymbol{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\right) - \mu_2\left(\boldsymbol{V}_0 + \boldsymbol{\Lambda}(\vec{x})\boldsymbol{V}_0\boldsymbol{M}_{1,0}\right) - \lambda\boldsymbol{M}_{1,0}(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{P}^{\geq}) \\ & - \lambda\left(\boldsymbol{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\right)\boldsymbol{P}^{\geq}\boldsymbol{B} - \lambda a_1\boldsymbol{M}_{1,0}\boldsymbol{P}^{=} - \lambda a_2\boldsymbol{\Lambda}(\vec{y})(\boldsymbol{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\ )\ \boldsymbol{P}^{=}\boldsymbol{D}\end{aligned}$$

Similarly, for the vector representation $\vec{f}$ of $F(x,y,s)$, we obtain

$$\boldsymbol{\Gamma}_F^{-1}(s)\vec{f} = \mu_2\vec{c} \tag{7}$$

where $\vec{c}$ is given by Eq. (5) as before and

$$\begin{aligned}\boldsymbol{\Gamma}_F^{-1}(s) =& \boldsymbol{\Lambda}(s+\lambda+(1-\vec{x})\mu_1+(1-\vec{y})\mu_2)\left(\boldsymbol{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\right) - \mu_1\left(\boldsymbol{U}_0 + \boldsymbol{\Lambda}(\vec{y})\boldsymbol{U}_0\boldsymbol{M}_{0,1}\right) - \lambda\boldsymbol{M}_{0,1}(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{P}^{\leq}) \\ & - \lambda\left(\boldsymbol{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\right)\boldsymbol{P}^{\leq}\boldsymbol{A} - \lambda a_2\boldsymbol{M}_{0,1}\boldsymbol{P}^{=} - \lambda a_1\boldsymbol{\Lambda}(\vec{x})(\boldsymbol{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\ )\ \boldsymbol{P}^{=}\boldsymbol{C}\end{aligned}$$

We can now approximate the functions $E(x,y,s)$ and $F(x,y,s)$ by an interpolation of their respective vector representations $\boldsymbol{\Gamma}_E(s)\mu_1\vec{c}$ and $\boldsymbol{\Gamma}_F(s)\mu_2\vec{c}$ over the discretization set pairs $\{(x_i,y_j) \mid 1 \le i,j \le n\}$.

3.2. Unconditional response time

Let $\vec{e}^{<} = (\boldsymbol{I} - \boldsymbol{P}^{\geq})\vec{e}$, $\vec{f}^{>} = (\boldsymbol{I} - \boldsymbol{P}^{\leq})\vec{f}$, $\vec{e}^{=} = \boldsymbol{P}^{=}\vec{e}$ and $\vec{f}^{=} = \boldsymbol{P}^{=}\vec{f}$. Then Eq. (3) for $W^*(s)$ in Proposition 2 can be transformed as follows, omitting s from the integrands for brevity:

$$\begin{aligned}W^*(s) \approx \frac{1}{(2\pi)^2}\Bigg(&\int_0^{2\pi}\int_0^{2\pi}\Im(\vec{e}^{<})(r_1e^{it_1}, r_1e^{it_2})\Im(\vec{g})(r_2e^{-it_1}, r_2e^{-it_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi}\Im(\vec{f}^{>})(r_1e^{it_1}, r_1e^{it_2})\Im(\vec{g})(r_2e^{-it_1}, r_2e^{-it_2})dt_1dt_2 \\ &+ \int_0^{2\pi}\int_0^{2\pi}a_1\Im(\vec{e}^{=})(r_1e^{it_1}, r_1e^{it_2})\Im(\vec{g})(r_2e^{-it_1}, r_2e^{-it_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi}a_2\Im(\vec{f}^{=})(r_1e^{it_1}, r_1e^{it_2})\Im(\vec{g})(r_2e^{-it_1}, r_2e^{-it_2})dt_1dt_2\Bigg)\end{aligned}$$

where $\mathfrak{I}(\vec{e}^{\sim})$, $\mathfrak{I}(\vec{f}^{\sim})$ and $\mathfrak{I}(\vec{g})$ are two-variable interpolations of the above defined $\vec{e}^{\sim}$, $\vec{f}^{\sim}$ and $\vec{g}$ vectors, respectively.

One can choose the contours on which the original discretization points lie to match the circles of this double integral that calculates $W^*(s)$ according to Eq. (3). That is, the circles used in the estimation of E and F have radius r_1 and the circles used to estimate the pgf G have radius $1/r_1$. This leads to a straightforward evaluation of the integral by interpolating the discretized values around the circles.

3.3. Moments

The kth moment of response time is $(-1)^k$ multiplied by the kth derivative of $W^*(s)$ evaluated at $s = 0$. This requires the partial derivatives of $E(x, y, s)$ and $F(x, y, s)$ wrt s at $s = 0$, which are determined below in Lemma 2. First, recall that the solution for the vector representation of $E(x, y, s)$ in Eq. (6) is $\vec{e} = \boldsymbol{\Gamma}_E(s)\mu_1\vec{c}$. The results for $F(x, y, s)$ are similar.

Lemma 2. *The kth partial derivative of $E(x, y, s)$ wrt s at $s = 0$ has vector representation $\vec{e}_k$ given by the recurrence:*

$$\vec{e}_0 = \boldsymbol{\Gamma}_E(0)\mu_1\vec{c}$$
$$\vec{e}_k = -k\boldsymbol{\Gamma}_E(0)\left(\vec{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\right)\vec{e}_{k-1}$$

where $\vec{c}$ is given by Eq. (5) and $\boldsymbol{\Gamma}_E(0)$ is defined by Eq. (7) with the substitution $s = 0$.

Similarly, the kth partial derivative of $F(x, y, s)$ wrt s at $s = 0$ has vector representation $\vec{f}_k$ given by the recurrence:

$$\vec{f}_0 = \boldsymbol{\Gamma}_F(0)\mu_2\vec{c}$$
$$\vec{f}_k = -k\boldsymbol{\Gamma}_F(0)\left(\vec{I}^{n^2\times n^2} + \boldsymbol{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\right)\vec{f}_{k-1}$$

where $\vec{c}$ is the same as before and $\boldsymbol{\Gamma}_F(0)$ is defined by above with the substitution $s = 0$.

Proof. First, as already noted, $\vec{e}_0 = \boldsymbol{\Gamma}_E(0)\mu_1\vec{c}$. Differentiating Eq. (1) k times wrt s (denoted by the superscript (k)), noting that the variable s appears in only one coefficient and that the right-hand side is constant, yields, for $k \geq 1$,

$$\begin{aligned}
&\left(s + \lambda + (1-x)\mu_1 + (1-y)\mu_2\right)\left(E^{(k)}(x,y,s) + x\frac{\partial E^{(k)}}{\partial x}(x,y,s)\right) - \mu_2\left(E^{(k)}(x,0,s) + x\frac{\partial E^{(k)}}{\partial x}(x,0,s)\right) - \lambda\frac{\partial E^{(k)<}}{\partial x}(x,y,s)\\
&\quad - \lambda\left(\beta_E^{(k)\geq}(x,y,s) + x\frac{\partial \beta_E^{(k)\geq}}{\partial x}(x,y,s)\right) - \lambda a_1\frac{\partial E^{(k)=}}{\partial x}(x,y,s) - \lambda a_2 y\left(\delta_E^{(k)=}(x,y,s) + x\frac{\partial \delta_E^{(k)=}}{\partial x}(x,y,s)\right)\\
&\quad = -k\left(E^{(k-1)}(x,y,s) + x\frac{\partial E^{(k-1)}}{\partial x}(x,y,s)\right)
\end{aligned}$$

The left-hand side is the left-hand side of (1) with E replaced by $E^{(k)}$. Hence, if we replace $\vec{c}$ with

$$\vec{c}_k = \left(-k\left(E^{(k-1)}(x_i,y_j,0) + x_i\frac{\partial E^{(k-1)}}{\partial x}(x_i,y_j,0)\right) \mid 1 \leq i \leq n, 1 \leq j \leq n\right)$$

we find that $\vec{e}_k = \boldsymbol{\Gamma}_E(0)\vec{c}_k$. The result then follows for $\vec{e}_k$ using the operators in Table 1. The proof for $\vec{f}_k$ is similar.

We note that there is no need to recalculate the inverse of a matrix for each moment; the same inverse matrix $\boldsymbol{\Gamma}_E(0)$ or $\boldsymbol{\Gamma}_F(0)$ is all that is needed. As a result, once these inverses have been computed, any number of moments can be calculated quickly. The unconditional response time moments are obtained by deconditioning according to Proposition 2, replacing $W^*(0)$ by the kth moment M_k, E by $E^{(k)}$ and F by $F^{(k)}$; actually, for $k = 0$, $W^*(0) = M_0 = 1$, providing an accuracy check.

4. Evaluation

In order to evaluate the methodology presented above for estimating the response time distribution of two JSQ-PS queues, we perform several experiments. First, the numerical response time moments and density are obtained for two networks, one with a medium load and one with a high load. The results are then compared against simulation. Next, JSQ scheduling is compared against static scheduling methods. This is followed by a section devoted to finding the optimal value of a_1, the probability of a customer joining the first queue given that the queues are of equal length. Finally, the algorithm's accuracy and computational cost are considered, and an optimized version of the algorithm is presented to further enhance its accuracy.

4.1. Comparison with simulation

We first evaluated our model by calculating response time densities for two sets of parameters. These parameters were chosen to test the model in both a medium and a heavy load scenario. The expectation is to have higher accuracy in the case of medium load. We inverted the Laplace transforms numerically using the Fixed Talbot inversion method to obtain the probability density functions [12].

For the medium load scenario, the parameters used are $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$, giving 50% utilization. Notice that in this case all arrivals are directed to the faster server when the queue lengths are equal; more details are given regarding the

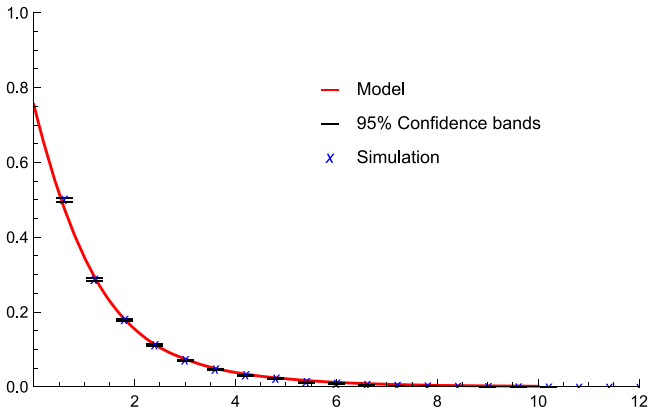

Fig. 3. Response time density $w(t)$ (vertical axis) against t (horizontal axis) for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$: comparison with simulation.

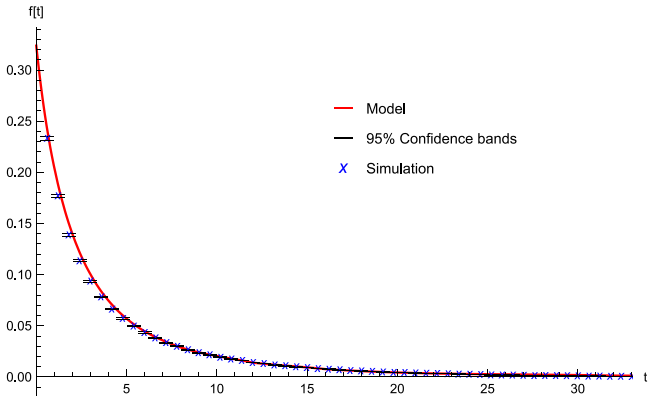

Fig. 4. Response time density $w(t)$ (vertical axis) against t (horizontal axis) for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$: comparison with simulation.

particular choice of routing probabilities a_1, a_2 in Section 4.3. We used 64 points around the circle for both coordinates when solving for E and F. The density obtained was then plotted against corresponding simulation results. For the latter, we used regenerative simulation with 500,000 regeneration cycles; see [13]. Following [14], these sets of cycles were partitioned into batches, the sizes of which are chosen equal to the square root of the number of cycles, or nearest integer; here 707, giving a total of 707 batches as well. Fig. 3 shows that the model's accuracy is remarkable in this case.

For the heavy traffic scenario, the parameters used are $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$, so either queue would be saturated on its own since $\lambda > \mu_2 > \mu_1$; the scheduling strategy is therefore more critical. As before, we compared the results with the corresponding simulation. The densities are shown in Fig. 4 and still show good agreement. Note, that the numbers of points used to calculate E and F were increased from 64 to 120 to achieve sufficient accuracy.

Finally, we used Lemma 2 to compute response time moments up to the fourth by differentiation of $W^*(s)$ at $s = 0$. This is much more efficient than calculating the response time densities as it only requires the model to run for a single s-value, namely $s = 0$. This is in contrast to estimating the density, where, at each time-point t, we need to calculate $W^*(s)$ for at least 16 s-values in order to be able to invert the LST with reasonable accuracy. Table 3 compares the first four moments of response time to those obtained from the corresponding regenerative simulation in the case of heavy traffic. The model results show very good accuracy for higher moments, well within the 95% confidence intervals, and the first moment just outside. Moments for the medium load scenario along with a detailed discussion on optimal routing probabilities are given in Section 4.3.

4.2. Static scheduling models

It is interesting to compare this JSQ model with other, static scheduling policies for the same two servers with total arrival rate λ. The obvious policy would be to split the arrival stream so that each queue gets arrivals at a rate proportional to its service rate; i.e. arrival rates $p_1\lambda$ and $p_2\lambda$ to queue 1 and queue 2, respectively, where $p_1 = \mu_1/(\mu_1 + \mu_2)$ and $p_2 = \mu_2/(\mu_1 + \mu_2)$. The two queues then have known response time densities [15]. Notice that not any choice of probabilities p_1, p_2 give equilibrium, which requires

Table 3
Response time moments for two JSQ-PS queues with parameters $\lambda = 1$, $\mu_1 = 0.4$, $\mu_2 = 0.8$, $a_1 = a_2 = 0.5$.

Moment	JSQ/PS model	Simulation	95% CB
1st	6.15533	6.0914	±0.022648
2nd	125.622	124.991	±1.31098
3rd	5472.74	5547.67	±128.633
4th	403 965	422 517	±19382.9

Table 4
Comparison of response time moments for JSQ/PS, JSQ/FCFS, static load balanced with PS and static load balanced with FCFS; parameters: $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$.

Moment	JSQ/PS	JSQ/FCFS	Static/PS	Static/FCFS
1st	6.15533	6.07415	10.0	10.0
2nd	125.622	79.3979	385.714	225.0
3rd	5472.74	1678.18	35 127.6	8437.5
4th	403 965	49 942	5 862 020	455 625

Table 5
Moments as the probability, a_1, of joining queue 1 at equal queue lengths varies; model parameters: $\lambda = 2, \mu_1 = 1, \mu_2 = 3$.

Prob. of joining queue 1, a_1	Moments (in the form: *model [simulation]*)			
0.0	0.731 [0.728]	1.638 [1.637]	7.455 [7.466]	53.204 [52.871]
0.1	0.761 [0.757]	1.787 [1.777]	8.461 [8.439]	62.185 [62.735]
0.25	0.813 [0.798]	2.038 [1.983]	10.099 [9.817]	76.474 [74.304]
0.5	0.865 [0.858]	2.360 [2.319]	12.604 [12.228]	101.432 [96.921]
0.75	0.920 [0.911]	2.694 [2.643]	15.215 [14.746]	127.729 [121.712]
1.0	0.969 [0.957]	3.009 [2.931]	17.794 [17.045]	154.705 [145.464]

$p_1\lambda < \mu_1$ and $p_2\lambda < \mu_2$; i.e. $1 - \mu_2/\lambda < p_1 < \mu_1/\lambda$. Hence in our higher load case, we require $0.2 < p_1 < 0.4$. The load balanced choice above has $p_1 = \mu_1/(\mu_1 + \mu_2) = 1/3$, $p_2 = 2/3$ and yields equal utilization 0.8333 at the two queues. The response time in the static scheduling model has first four moments (obtained by differentiating the known Laplace transform of the density function) 10, 385.714, 35,127.6, and 5,862,020, very much higher than those achieved by JSQ scheduling (see Table 4).

4.3. Finding the optimal routing probabilities

In this section, our goal is to find the value of a_1 – the probability of an arriving task joining queue 1 given that the queues have equal length – that produces the best performance, that is, one that corresponds to the lowest moments. In the case of two identical queues ($\mu_1 = \mu_2$), it is straightforward to see that the optimal routing is $a_1 = a_2 = 0.5$, otherwise, one of the servers becomes slightly overloaded. However, when the service rates are different, $a_1 = a_2 = 0.5$ is not expected to be optimal. One obvious candidate to try is splitting arrivals in proportion to each queue's service rate, similar to the static scheduling model above, making $a_1 = \frac{\mu_1}{\mu_1+\mu_2}$.

The set of parameters used for this experiment is $\lambda = 2, \mu_1 = 1, \mu_2 = 3$, and we take several values of a_1 between 0 and 1. The results are displayed in Table 5. The rows of Table 5 correspond to different values of a_1 and the moments computed using simulation are in brackets right next to their model-based counterpart. Before focusing on the optimal row of Table 5, one observation to make is that the moments calculated by the model are extremely accurate.

The a_1 value corresponding to the lowest moments is $a_1 = 0$. This suggests, perhaps not surprisingly, that when the service rates differ widely – in the current example queue 2 serves customers three times as fast as queue 1 – the optimal strategy is to send everything to the faster server when there is a choice.

We reran the experiments with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$ to see if this is still the case when the service rates are much closer together; now queue 2 is only slightly more efficient than queue 1. Table 6 shows the results. Again, the moments calculated by the model remain spot on.

The first row corresponding to $a_1 = 0$ remains the one with the lowest moments, suggesting that sending everything to the faster server is still the optimal strategy, even when service rates differ ever so slightly.

These findings prompt us to think about variations of JSQ scheduling itself. One can imagine a scheduling similar to JSQ where, given $\mu_1 < \mu_2$, arriving customers go to queue 1 if $i < j - 1$ and to queue 2 if $i > j - 1$, and routing probabilities are used to break ties when $i = j - 1$. More generally, we can shift the difference in the number of jobs by k, and route customers to the slower server when its queue is k customers shorter than the faster server's queue. A natural question to ask is what is the optimal value of k for a given pair of service rates.

Table 6
Moments as the probability, a_1, of joining queue 1 at equal queue lengths varies; parameters: $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$.

Prob. of joining queue 1, a_1	Moments (in the form: *model [simulation]*)			
0.0	1.399 [1.396]	4.584 [4.574]	25.778 [25.690]	215.724 [214.135]
0.2	1.406 [1.407]	4.640 [4.630]	26.332 [26.079]	222.872 [218.336]
0.4	1.434 [1.427]	4.818 [4.773]	27.765 [27.361]	238.088 [232.156]
0.5	1.443 [1.431]	4.883 [4.795]	28.371 [27.468]	245.269 [232.094]
0.6	1.452 [1.443]	4.951 [4.902]	29.019 [28.872]	253.074 [257.151]
0.8	1.469 [1.457]	5.094 [4.998]	30.43 [29.414]	270.534 [258.434]
1.0	1.486 [1.478]	5.247 [5.199]	32.021 [31.501]	290.419 [280.881]

4.4. Accuracy and cost

As per [7], we know the worst-case errors for each of the matrices given in Table 1. The worst-case errors of Table 2 are covered by the same propositions, since using Proposition 3, the matrices can be expressed as linear combinations of matrices in Table 1. The operation with the largest asymptotic error corresponds to calculating derivatives with a worst-case error of $\Theta(n^{-1})$ for each element of the matrix. Once the functional equations are transformed into a system of linear equations, the matrix on the left-hand side is inverted. In our numerical calculations, the condition number of this matrix is low so that the error of the inverted matrix remains of the same order. Finally, the inverted matrix is multiplied with a vector that is known exactly. Therefore, the error of the sum is of order $\sqrt{n}\Theta(n^{-1}) = \Theta(n^{-1/2})$ where the multiple of $\sqrt{n}$ comes from the Central Limit Theorem and $\Theta(n^{-1})$ is the individual error of each term. However, we found the error to be around 10–100 times smaller in practice; this is unsurprising because the above error estimates correspond to the worst-case scenario, not the average.

Regarding the cost of the algorithm, each of the matrices in Tables 1 and 2, corresponding to the operators in Eqs. (6) and (7), has size $n^2 \times n^2$ and is calculated element-wise. Therefore, each requires storage and has computation time of order $\mathcal{O}(n^4)$. We used approximately 100 points in each contour, so this is in the region of 10^8. However, these constant matrices need only be calculated once for any given n and then can be stored and reused. To get the final equations of form $\boldsymbol{A}(s)\vec{x} = \vec{b}$, a linear combination of these individual matrices needs to be calculated, which depends on s, but these calculations are much faster than the calculations of the matrices in Table 1. However, for every time-point t in the density function, we need approximately $16 - 32$ values of s, and corresponding matrices $\boldsymbol{A}(s)$, to be able to invert the Laplace transform numerically. The product of 32 and the number of time points required to approximate the density function well is significantly larger than n in our calculations, so the overall cost exceeds $\Theta(n^5)$. Next, the linear equations need to be solved for each s-point, which requires the inversion of the matrices $\boldsymbol{A}(s)$ – a cubic operation in the dimension of the input matrix, in our case giving $\mathcal{O}(n^6)$ cost. This is the dominating factor, but in practice, we found the time needed to be of the same order as calculating the $\boldsymbol{A}(s)$ matrices. This may be partly because $\boldsymbol{A}(s)$ requires the calculation of several matrices that each take $\Theta(n^5)$ time. The remaining operations, e.g., inverting the Laplace transforms to obtain the densities and the calculation of moments, are negligible in comparison. Optimization of these calculations is possible and would likely decrease the computational cost several-fold. Finally, note that the corresponding simulation results have run for several days.

5. Optimizations

In this section two possible optimization methods are considered to increase the accuracy of our algorithm.

5.1. Optimization I: inversion of operator order

The first optimization inverts the order of some of the operators in Eqs. (1) and (2). These equations have several terms involving a derivative of a partial generating function. We evaluate the efficacy of performing the differentiation first and then transforming the resulting generating functions into partial versions. This could improve accuracy as the generating functions are Laplace transforms — smooth functions whose derivatives, when approximated, are likely to retain high precision. In fact when $s = 0$, as in the calculation of moments, the Laplace transforms are the constant function 1, with easily approximated derivative 0 everywhere. To facilitate this inversion, we need the following auxiliary lemma.

Lemma 3. *If $H(x, y)$ is a two-parameter function that is analytic for all x, y with $|x| \le r_1, |y| \le r_2$, then for any x_0, y_0 with $|x_0| = r_1$ and $|y_0| = r_2$:*

$$H^{>}(x_0, y_0) = -\frac{1}{2\pi i}\oint_{C_y} \frac{H(x_0 y_0/y, y)}{y - y_0} dy \tag{8}$$

and

$$H^{<}(x_0, y_0) = -\frac{1}{2\pi i}\oint_{C_x} \frac{H(x, x_0 y_0/x)}{x - x_0} dx \tag{9}$$

where the contours C_x, C_y are, respectively, circles centred on the origin and lying in the annuli of analyticity of $H(x, x_0 y_0/x)$, $H(x_0 y_0/y, y)$, with radii less than r_1, r_2.

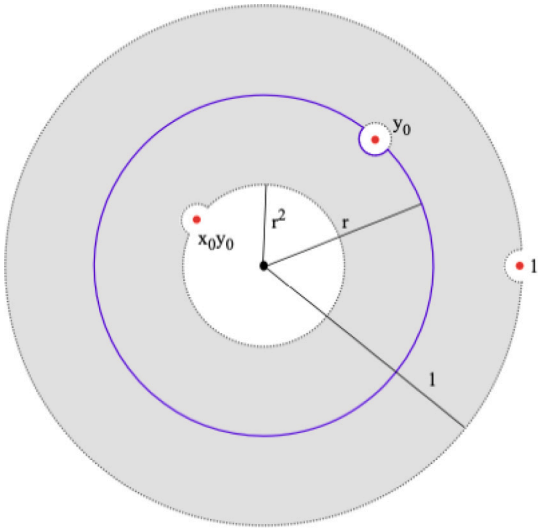

Fig. 5. The contour of integration (in blue) – a distorted circle going around y_0 on the inner side; and the area of convergence (in grey) – a punctured annulus with inner radius r^2 and outer radius 1. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Proof. Let $H(x,y)=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_{ij}x^iy^j$ with radii of convergence R_x, R_y, so that $r_1<R_x$ and $r_2<R_y$. Then the annulus of analyticity of $H(x_0y_0/y,y)$ is the region between the circles $D(0,r_1r_2/R_x)$ and $D(0,r_2)$. Noting that $|y|<|y_0|$ for $y\in C_y$:

$$\oint_{C_y}\frac{H(x_0y_0/y,y)}{y-y_0}dy=-\oint_{C_y}\frac{H(x_0y_0/y,y)}{y_0}\frac{1}{1-y/y_0}dy=-\oint_{C_y}\left[\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_{ij}y_0^{-1}(x_0y_0)^iy^{j-i}\right]\left[\sum_{n=0}^{\infty}(y/y_0)^n\right]dy$$
$$=-\oint_{C_y}\left[\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{n=0}^{\infty}a_{ij}y_0^{-1}(x_0y_0)^iy^{j-i+n}y_0^{-n}\right]dy=-\oint_{C_y}\left[\sum_{i=0}^{\infty}\sum_{j=0}^{i-1}a_{ij}y_0^{-1}(x_0y_0)^iy^{-1}y_0^{j+1-i}\right]dy=-2\pi iH^{>}(x_0,y_0)$$

Eq. (9) follows by symmetry.

Remark. For Eq. (8) (respectively (9)), the integration contour lies in the annulus of convergence of $H(x_0y_0/y,y)$ (respectively $H(x,x_0y_0/x)$) and encircles the origin, passing inside the point y_0. Since $H(x_0y_0/y,y)$ and $H(x,x_0y_0/x)$ are analytic in the annuli of convergence, the Cauchy-integral around a small closed contour γ, say, oriented counter-clockwise, that encloses the points y_0, x_0 in the respective cases, each evaluates to $2\pi iH(x_0,y_0)$. Consequently, since $H^{\leq}(x_0,y_0)+H^{>}(x_0,y_0)=H(x_0,y_0)$, application of Cauchy's integral theorem to the counter-clockwise oriented "inner" contour C_y joined onto γ, which converts C_y into an "outer" contour, yields the result by application of Lemma 2 of [7].

Corollary 1. *Lemma 3 extends to the case where the contours C_x, C_y are, respectively, the circles centred on the origin with radii r_1, r_2, distorted to go around the points x_0, y_0 on the inner side.*[1]

The proof follows by continuity since in the region $D(0,r_1)\times D(0,r_2)$ and on its boundaries ($x=r_1$ or $y=r_2$), $H(x,y)$ is analytic. The following result now provides the necessary formulas for inverting the order of differentiation and partialization.

Proposition 4. *If $H(x,y)$ is a two-parameter function that is analytic for all x,y with $|x|\leq r, |y|\leq r$, then for any x_0, y_0 with $|x_0|=|y_0|=r$*

$$\frac{\partial}{\partial x}H^{>}(x_0,y_0)=\left(\frac{\partial}{\partial x}H(x_0,y_0)\right)^{\geq}\qquad\frac{\partial}{\partial x}H^{\leq}(x_0,y_0)=\left(\frac{\partial}{\partial x}H(x_0,y_0)\right)^{<}$$
$$\frac{\partial}{\partial x}H^{=}(x_0,y_0)=\frac{1}{x_0}\left(x_0\frac{\partial}{\partial x}H(x_0,y_0)\right)^{=}$$

$\frac{\partial}{\partial x}H^{\geq}(x_0,y_0)$ and $\frac{\partial}{\partial x}H^{<}(x_0,y_0)$ can be derived from the above, using the linearity of differentiation, e.g., $\frac{\partial}{\partial x}H^{\geq}(x_0,y_0)=\frac{\partial}{\partial x}H^{>}(x_0,y_0)+\frac{\partial}{\partial x}H^{=}(x_0,y_0)$.

[1] i.e. the distortions (typically semi-circles with vanishing radius) traverse the point on the side towards the origin.

Proof. Using Lemma 3 and its corollary, we have

$$\frac{\partial}{\partial x}H^{>}(x_0,y_0)=\frac{\partial}{\partial x}\left(-\frac{1}{2\pi i}\oint_{C_y}\frac{H(x_0y_0/y,y)}{y-y_0}dy\right)=-\frac{1}{2\pi i}\oint_{C_y}\frac{\frac{\partial}{\partial x}H(x_0y_0/y,y)}{y-y_0}\frac{y_0}{y}dy$$

where the contour of integral C_y shown in Fig. 5 goes around y_0 on the inner side. Now, let $x = x_0y_0/y$ and note that this change of variable transforms C_y into a clockwise contour C_x that goes around x_0 on the outside. Therefore, we get

$$\frac{1}{2\pi i}\oint_{C_x}\frac{\frac{\partial}{\partial x}H(x,x_0y_0/x)}{x_0y_0/x-y_0}\frac{y_0x}{x_0y_0}\left(-\frac{x_0y_0}{x^2}\right)dx=\frac{1}{2\pi i}\oint_{C_x}\frac{\frac{\partial}{\partial x}H(x,x_0y_0/x)}{x-x_0}dx$$

This is exactly the integral given by [7, Lemma 2] for $\left(\frac{\partial}{\partial x}H(x_0,y_0)\right)^{\geq}$. The other two equations can be proved following similar steps.

Similar equations can be derived for partial derivatives with respect to y_0 by symmetry. In matrix form, Proposition 4 is applied by, for example, substituting $\boldsymbol{M}_{1,0}\boldsymbol{P}^{=}$ with $\boldsymbol{\Lambda}(1/\vec{x})\boldsymbol{P}^{=}\boldsymbol{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}$ in the definition of $\boldsymbol{\Gamma}_E(s)$ in Eq. (7).

5.2. Optimization II – simplification of the partialization operator

When $s = 0$, the LST of the sojourn time of a task at queue 1, $U^*_{i,j}(s)$, simplifies to the integral of the density function of the random variable $U_{i,j}$ and therefore $U^*_{i,j}(0) = 1$. Furthermore, for $\Re(s) > 0$, $|U^*_{i,j}(s)| < 1$, for all $i,j \geq 0$ as $|U^*_{i,j}(s)| \leq \int_0^\infty f(t)|e^{-st}|dt = \int_0^\infty f(t)e^{-\Re(s)t}dt < 1$; of course the same applies to $V^*_{i,j}(s)$. Thus, $E(x,y,0) = F(x,y,0) = \frac{1}{(1-x)(1-y)}$ and for $\Re(s) > 0$, $|E(x,y,s)| < \frac{1}{(1-|x|)(1-|y|)}$. We define $\tilde{E}(x,y,s) = (1-x)(1-y)E(x,y,s)$ and $\tilde{F}(x,y,s) = (1-x)(1-y)F(x,y,s)$. Then we have $\tilde{E}(x,y,0) = \tilde{F}(x,y,0) = 1$. These constant functions are numerically more desirable than $E(x,y,s)$ and $F(x,y,s)$ with their sharp peaks near $x = 1$ and $y = 1$. This approach is motivated by the improvements seen in [7, Section 4.2] where similar transformations greatly improved the accuracy of the obtained response time distribution for a tandem pair of PS and FCFS queues.

Note that $\tilde{E}(x,y,s) = (1-x)(1-y)E(x,y,s)$ is a linear transformation of $E(x,y,s)$. In vector-matrix notation, $\vec{\tilde{e}} = \boldsymbol{\Lambda}((1-\vec{x})(1-\vec{y}))\vec{e}$. This defines a linear map $\mathcal{V} \to \widetilde{\mathcal{V}}$ on the vector space $\mathcal{V}$ of complex vectors of length mn. Then, each operator matrix $\boldsymbol{M}$ of (6) and (7) in $\mathcal{V}$ is mapped to the matrix $\widetilde{\boldsymbol{M}} = \boldsymbol{\Lambda}((1-\vec{x})(1-\vec{y}))\boldsymbol{M}\boldsymbol{\Lambda}(\frac{1}{(1-\vec{x})(1-\vec{y})})$ in $\widetilde{\mathcal{V}}$. The equation we solve in $\widetilde{\mathcal{V}}$, with mapped vectors and matrices, remains the same. Our aim now is to calculate and simplify the matrices for some of the operations in the new vector space directly. For differentiation and evaluation at $x = 0$ or $y = 0$, this is straightforward. For example,

$$\widetilde{\boldsymbol{M}}_{1,0}=\boldsymbol{\Lambda}((1-\vec{x})(1-\vec{y}))\boldsymbol{M}_{1,0}\boldsymbol{\Lambda}\left(\frac{1}{(1-\vec{x})(1-\vec{y})}\right)=\boldsymbol{\Lambda}\left(\frac{1}{1-\vec{x}}\right)+\boldsymbol{M}_{1,0}$$

and

$$\widetilde{\boldsymbol{U}_0}=\boldsymbol{\Lambda}((1-\vec{x})(1-\vec{y}))\boldsymbol{U}_0\boldsymbol{\Lambda}\left(\frac{1}{(1-\vec{x})(1-\vec{y})}\right)=\boldsymbol{\Lambda}(1-\vec{y})\boldsymbol{U}_0$$

In order to calculate the matrices for partialization in the new vector space, we derive equations in the following lemma that relate the partialized versions of generating function $\tilde{E}$ to those of E.

Lemma 4. *Let $r = r_1 = r_2 < 1$ and $|x_0| = |y_0| = r$. Then*

$$E^{>}(x_0,y_0)=\frac{x_0\tilde{E}^{\leq}(1,x_0y_0)}{(x_0-1)(x_0y_0-1)}+\frac{\tilde{E}^{>}(x_0,y_0)}{(x_0-1)(y_0-1)}-\frac{\tilde{E}^{>}(x_0y_0,1)}{(y_0-1)(x_0y_0-1)} \tag{10}$$

$$E^{<}(x_0,y_0)=\frac{y_0\tilde{E}^{\geq}(x_0y_0,1)}{(y_0-1)(x_0y_0-1)}+\frac{\tilde{E}^{<}(x_0,y_0)}{(x_0-1)(y_0-1)}-\frac{\tilde{E}^{<}(1,x_0y_0)}{(x_0-1)(x_0y_0-1)} \tag{11}$$

$$E^{=}(x_0,y_0)=\frac{\tilde{E}^{=}(x_0,y_0)+\tilde{E}^{>}(x_0y_0,1)+\tilde{E}^{<}(1,x_0y_0)}{1-x_0y_0} \tag{12}$$

Proof. Let C_y be the circle centred on the origin with radius r_2, distorted to go round y_0 on the inner side as shown in Fig. 5. Using Lemma 3 and its corollary, we have

Table 7
Comparison of the optimized method with the original algorithm for calculating response time moments (model parameters: $\lambda = 2, \mu_1 = 1, \mu_2 = 3$ and $a_1 = 0.5$.).

Moment	JSQ/PS model	Opt. I	Opt. I + II	Simulation
1st	0.865	0.863	0.862	0.858
2nd	2.360	2.343	2.340	2.319
3rd	12.604	12.447	12.438	12.228
4th	101.432	99.603	99.648	96.921

Table 8
Comparison of the optimized method with the original algorithm for calculating response time moments (model parameters: $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$ and $a_1 = 0.5$.).

Moment	JSQ/PS model	Opt. I	Opt. I + II	Simulation
1st	1.443	1.441	1.44	1.431
2nd	4.883	4.871	4.866	4.795
3rd	28.371	28.266	28.228	27.468
4th	245.269	244.102	243.676	232.094

Table 9
Comparison of the optimized method with the original algorithm for calculating response time moments (model parameters: $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8$ and $a_1 = 0.5$.).

Moment	JSQ/PS	Opt. I	Opt. I + II	Simulation
1st	6.155	6.127	6.10848	6.091
2nd	125.622	126.327	125.966	124.991
3rd	5472.74	5597.28	5629.87	5547.67
4th	403 965	423 419	434 455	422 517

$$E^{>}(x_0, y_0) = \frac{1}{2\pi i} \oint_{C_y} \frac{\tilde{E}(x_0y_0/y, y)}{(1 - x_0y_0/y)(y-1)(y-y_0)} dy = \frac{1}{2\pi i} \oint_{C_y} \frac{y\tilde{E}(x_0y_0/y, y)}{(y - x_0y_0)(y-1)(y-y_0)} dy$$

Noting that y_0, x_0y_0 and 1 are three distinct values, we can decompose the integrand into partial fractions. Omitting the repeating $\tilde{E}(x_0y_0/y, y)$ factors for brevity, we have:

$$\frac{y}{(y - x_0y_0)(y-1)(y-y_0)} = \frac{x_0}{(x_0-1)(x_0y_0-1)} \frac{1}{y - x_0y_0} - \frac{1}{(x_0-1)(y_0-1)} \frac{1}{y-y_0} + \frac{1}{(y_0-1)(x_0y_0-1)} \frac{1}{y-1}$$

The first of the ensuing three integrals is given by Lemma 2 of [7] as x_0y_0 lies inside the contour C_y, whereas the other two integrals are calculated using Lemma 3 as y_0 and 1 both lie outside. Hence

$$\frac{1}{2\pi i} \oint_{C_y} \frac{y\tilde{E}(x_0y_0/y, y)}{(y - x_0y_0)(y-1)(y-y_0)} dy = \frac{x_0}{(x_0-1)(x_0y_0-1)} \tilde{E}^{\leq}(1, x_0y_0) + \frac{1}{(x_0-1)(y_0-1)} \tilde{E}^{>}(x_0, y_0) - \frac{1}{(y_0-1)(x_0y_0-1)} \tilde{E}^{>}(x_0y_0, 1)$$

Eq. (11) follows by symmetry and Eq. (12) by using $E^{=} = E - E^{>} - E^{<}$, noting that $E^{=}(x_0, y_0) = E^{=}(1, x_0y_0) = E^{=}(x_0y_0, 1)$.

Using Lemma 4, matrix $\boldsymbol{P}^{=}$ in the new vector space, for example, becomes

$$\widetilde{\boldsymbol{P}}^{=} = \boldsymbol{\Lambda}((1-\vec{x})(1-\vec{y}))\boldsymbol{P}^{=}\boldsymbol{\Lambda}\left(\frac{1}{(1-\vec{x})(1-\vec{y})}\right) = \boldsymbol{\Lambda}\left(\frac{(1-\vec{x})(\vec{1}-\vec{y})}{1-\vec{x}\vec{y}}\right)(\boldsymbol{P}^{=} + \boldsymbol{P}^{>}_{xy,1} + \boldsymbol{P}^{<}_{1,xy})$$

where $\boldsymbol{P}^{>}_{xy,1}$ is the matrix corresponding to the linear approximation of the operation $H^{>}(x_0y_0, 1) = -\frac{1}{2\pi i}\int_{C_y} \frac{H(x_0y_0/y,y)}{y-1} dy$.

5.3. Numerical experiments

We evaluated the optimizations presented above on three queueing models: two operating under medium load (50% utilization), as shown in Tables 7 and 8, and one with heavy load (~83% utilization), displayed in Table 9. At medium load, we found 64 discretization points to be sufficient on both the x- and y-contours, but for the high load parameterization 64 points gave poor accuracy and we used 120. For all three models, two rounds of testing were carried out. In the first round, Optimization I was run by itself. The results are shown in the third column of Tables 7–9. There are improvements in accuracy across all moments compared to the original algorithm, achieved without any increase in the running time. It is important to note, however, that in the case of heavy load, the simulation results – especially the higher moments – are prone to error, and we believe that the moments calculated by our model provide more reliable estimates.

Motivated by these findings, we ran Optimizations I and II together in the second round of testing. The results are displayed in the fourth column of the tables. While combining the two optimization methods further enhances the accuracy of the results (with occasional exceptions, such as the fourth moment in Table 7, which is slightly less accurate), this improvement comes at the cost of a significant increase in running time. Moreover, the gains compared to using Optimization I alone are relatively minor. Therefore, we conclude that Optimization I is the most effective approach.

```
interpolation2[alx_, aly_, tabv_, {o1_, o2_}] := Module[{ip1j, ip2i}, ip1j = Interpolation[Thread[List[alx, tabv[[All, #]]]], InterpolationOrder → o1] &;
  Function[x, ip2i = Interpolation[Thread[List[aly, ((ip1j[#][x]) & /@ Range[Length[aly]])]], InterpolationOrder → o2]][#1][#2] &];
```

Fig. A.6. Mathematica code for 2-dimensional linear interpolation.

6. Conclusion

JSQ-PS is a widely used scheduling policy but its response time density – even in the simplest case of two queues – has not been obtained up until now. We have used a novel numerical approximation technique to obtain the response time moments and density function for a pair of JSQ queues with PS discipline. The steps are as follows. First, functional equations for the generating functions E and F of conditional response times are derived, where the conditioning is on whether the arriving tagged task joins the first or the second queue. These functional equations are then solved numerically by transforming them into linear algebra. The generating function G of the initial equilibrium queue-length probabilities is available from [7]. Once the generating functions G, E, F are approximated, together with their modifications (e.g. $E^{\sim}$), the evaluation of a complex integral is required to express the LST of the unconditional response time distribution, from which the density itself is obtained by numerical inversion. Moments are obtained very efficiently in a straightforward manner by recursively calculating derivatives of $W^*(s)$ at $s = 0$. In Section 4, we demonstrated that our numerical method accurately computes the response time density and corresponding moments under both heavy and medium loads. Furthermore, in Section 5 we showed that changing the order of some of the operators and applying the transformation $\tilde{E}(x, y, s) = E(x, y, s)(1 - x)(1 - y)$, and a similar transformation for F, the accuracy of the algorithm is further improved.

Future work plans to explore variations on JSQ scheduling prompted by the experiments in Section 4.3 corresponding to finding the optimal values of a_1 and the offset parameter k. These variations can be thought of as shifted versions of the original JSQ scheduling policy, where the faster server automatically receives new arrivals until its queue is k customers longer than the slower server's queue. In addition to calculating the response time density for a shifted JSQ system, it is also interesting to try and find the optimal k value for a given set of service rates.

CRediT authorship contribution statement

Julianna Bor: Writing – review & editing, Methodology, Formal analysis, Validation, Investigation, Conceptualization. **Peter G. Harrison:** Writing – review & editing, Methodology, Formal analysis, Validation, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Mathematica code for two-dimensional interpolation

See Fig. A.6.

References

[1] Cisco IOS server load balancing configuration guide, 2011, (Accessed 14 November 2023). URL https://shorturl.at/hpgqQ.
[2] Microsoft.Network applicationGateways, 2023, (Accessed 14 November 2023). URL https://shorturl.at/I2ivr.
[3] Virtual IPs and service proxies, 2023, (Accessed on November 14, 2023). URL https://kubernetes.io/docs/reference/networking/virtual-ips/.
[4] Application load balancer now supports least outstanding requests algorithm for load balancing requests, 2019, (Accessed 14 November 2023). URL https://shorturl.at/HkxZ4.
[5] J.F.C. Kingman, Two similar queues in parallel, Ann. Math. Stat. 32 (4) (1961) 1314–1323, http://dx.doi.org/10.1214/aoms/1177704869.
[6] L. Flatto, H.P. McKean, Two queues in parallel, Comm. Pure Appl. Math. 30 (2) (1977) 255–263, http://dx.doi.org/10.1002/cpa.3160300206.
[7] P.G. Harrison, On the numerical solution of functional equations with application to response time distributions, Appl. Math. Comput. 472 (2024) http://dx.doi.org/10.1016/j.amc.2024.128637.
[8] B.K. Asare, F.G. Foster, Conditional response times in the M/G/1 processor-sharing system, J. Appl. Probab. 20 (4) (1983).
[9] P.G. Harrison, J. Bor, Response time distribution in a tandem pair of queues with batch processing, J. ACM 68 (2021) http://dx.doi.org/10.1145/3448973.
[10] L. Hormander, An Introduction to Complex Analysis in Several Variables, Elsevier, 1973.
[11] H.A. Priestley, Introduction to Complex Analysis Second Edition, second ed., Oxford University Press, 2003.
[12] J. Abate, W. Whitt, Numerical inversion of Laplace transforms of probability distributions, ORSA J. Comp. 7 (1) (1995).
[13] P.W. Glynn, Simulation algorithms for regenerative processes, in: S.G. Henderson, B.L. Nelson (Eds.), in: Handbooks in Operations Research and Management Science, vol. 13, Elsevier, 2006, http://dx.doi.org/10.1016/S0927-0507(06)13016-9.
[14] C. Chien, Batch size selection for the batch means method, in: J. D.Tew, S. Manivannan, D. Sadowski, A.F. Seila (Eds.), Proceedings of the 1994 Winter Sim. Conf., 1995, pp. 345–352, http://dx.doi.org/10.1109/WSC.1994.717192.
[15] E.G. Coffman Jr., R.R. Muntz, H.F. Trotter, Waiting time distributions for processor-sharing systems, J. ACM 17 (1) (1970) http://dx.doi.org/10.1145/321556.321568.