

Dispatching fixed-sized jobs with multiple deadlines to parallel heterogeneous servers



Esa Hyytiä^{a,c,*}, Rhonda Righter^b, Olivier Bilenne^c, Xiaohu Wu^c

^a Department of Computer Science, University of Iceland, Iceland

^b Department of Industrial Engineering and Operations Research, UC Berkeley, United States

^c Department of Communications and Networking, Aalto University, Finland

ARTICLE INFO

Article history:

Available online 25 May 2017

Keywords:

Dispatching problem

Parallel computing

Deadlines

M/D/1

MDP

ABSTRACT

We study the M/D/1 queue when jobs have firm deadlines for waiting (or sojourn) time. If a deadline is not met, a job-specific deadline violation cost is incurred. We derive explicit value functions for this M/D/1 queue that enable the development of efficient cost-aware dispatching policies to parallel servers. The performance of the resulting dispatching policies is evaluated by means of simulations.

© 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the dispatching problem, each arriving job is routed to one of the available servers immediately upon arrival. Even though a single fast server would often be preferred, the parallel servers are needed to match increasing capacity demands. Moreover, short latency, in the absence of preemptive scheduling, requires parallel servers.

In this paper, we consider a cost structure based on (firm) deadlines. Each job has a certain deadline for the maximum waiting time it can tolerate. If this waiting time is exceeded, a deadline violation cost is incurred, but the job must still be served. This cost structure stems from quality-of-experience metrics, where customers observe a good service level whenever the waiting time is “short”, but as soon as a given customer-specific threshold is exceeded, the observed service quality drops. That is, the tails of the response time distribution are one of the most crucial performance measures [1]. Similarly, service level agreements (SLAs) are often defined in terms of acceptable waiting times [2].

This basic setting has been studied recently in [3] in the context of M/G/1 queues. However, the results given there are either asymptotic or in the form of differential equations. In contrast, here we derive exact closed-form expressions (that satisfy the aforementioned differential equations and asymptotic behavior). More specifically, the main contributions of this paper are the first exact results for the value function and admission cost for the M/D/1 queue subject to a general deadline-based cost structure. Even though the service times are assumed to be fixed, the deadlines and their violation costs can vary according to some probability distributions. Moreover, there can be multiple deadlines with added cost for each deadline that is violated.

The approach itself is general, and traditionally the objective is the minimization of the mean sojourn time (see, e.g., [4–6]), possibly combined with the energy consumption (see, e.g., [7,8]). The value function for M/G/1-FCFS then enjoys elementary closed-form expressions. However, e.g., the processor sharing (PS) scheduling makes the situation more complex and exact results are available only for M/D/1-PS and M/M/1-PS [4,9]. The approach lends itself also to minimization of blocked jobs in loss systems [10].

* Corresponding author at: Department of Computer Science, University of Iceland, Iceland.
E-mail address: esa@hi.is (E. Hyytiä).

2. Basic model and notation

The basic model for a single M/D/1-FCFS queue with deadlines is as follows. We let λ denote the arrival rate and d the constant service time of a job in the M/D/1 queue so that the offered load is $\rho = \lambda d$. Jobs whose waiting time in queue, W , reach time τ , referred to as the deadline, incur a unit cost. This is equivalent to having a deadline $\tau + d$ for the sojourn time. We assume that $\rho < 1$ for stability, and the deadline must be positive to be meaningful, $\tau > 0$. The mean cost rate is

$$r = \lambda P\{W \geq \tau\}. \quad (1)$$

In general, the distribution of the waiting time cannot be expressed in simple terms, but instead in the form of the Laplace–Stieltjes Transform (LST) [11] or an infinite sum involving convolutions [12]. However, for the M/D/1 queue the waiting time distribution is available [13–15]

$$P\{W \leq \tau\} = (1 - \rho) \sum_{i=0}^{\lfloor \tau/d \rfloor} \frac{(\lambda(id - \tau))^i}{i!} e^{-\lambda(id - \tau)}. \quad (2)$$

In the general case, we have multiple classes of jobs, each with its own arrival rate λ_i , target deadline τ_i and i.i.d. deadline violation cost H_i . The total arrival rate is $\lambda = \sum_i \lambda_i$, and the stability requirement is that $\lambda d = \rho < 1$. The mean cost rate in this case is

$$r = \sum_i \lambda_i E[H_i] P\{W \geq \tau_i\}.$$

Our first task is to derive the so-called value function with respect to the deadline cost structure. Formally, the value function is defined as

$$v(u) \triangleq \lim_{t \rightarrow \infty} E[V(u, t) - rt],$$

where u is the current backlog (unfinished work) in the queue, and the random variable $V(u, t)$ denotes the deadline violation costs during time $(0, t)$ when the system is initially in state u . Given $\rho < 1$, the M/D/1 queue is stable, the system is ergodic, and the above limit is well-defined. (In fact, the limit is finite and well-defined also when $\rho \geq 1$ and the system is unstable.)

The M/G/1 queue has been analyzed in [3] in the context of the basic (single-class) cost structure. In particular, it is shown that the value function is a linear function of u for $u > \tau$, and for $0 \leq u \leq \tau$, $v(u)$ satisfies an integro-differential equation that can be solved numerically. Moreover, explicit results are given for M/G/1 when (i) $\tau < X$ and the load $\rho < 1$, and when (ii) $\tau \gg X$ and $\rho \rightarrow 1$ (the heavy-traffic regime), where X denotes the (random) service time. These two results naturally also hold for the corresponding M/D/1 queues.

In contrast, we analyze the general case when $\rho < 1$ and τ is arbitrary, and obtain an explicit closed-form expression for the value function. Moreover, we give the value function for the general multi-class case, and experiment with various dynamic dispatching policies obtained through one policy improvement step.

3. M/D/1 with single deadline

In this section, we assume a single deadline τ that applies to all jobs and a unit deadline violation cost, $h = 1$. These results are later generalized to multiple job classes with distinct deadlines in Section 4.

From [3], we know that the value function for $u > \tau$ is a linear function of u , i.e., $v(u) = v(\tau) + v_0(u)$ with

$$v_0(u) = \frac{\lambda - r}{1 - \rho}(u - \tau), \quad u > \tau. \quad (3)$$

In general, the value function satisfies the following differential equation,

$$v'(u) = -r + \lambda \mathbf{1}(u \geq \tau) + \lambda E[v(u + X) - v(u)], \quad u \geq 0, \quad (4)$$

where X denotes the random i.i.d. service time, and $\mathbf{1}(u \geq \tau)$ is 1 if the condition is true and otherwise zero. Additionally, $v'(0) = 0$. These equations can be solved numerically as discussed in [3]. However, exact closed-form results have not been available. For M/D/1, the differential equation (4) simplifies:

$$v'(u) + \lambda v(u) = -r + \lambda \mathbf{1}(u \geq \tau) + \lambda v(u + d), \quad u > 0. \quad (5)$$

In general, the mean cost rate r follows from the boundary condition $v'(0) = 0$. However, with M/D/1 we can use (2).

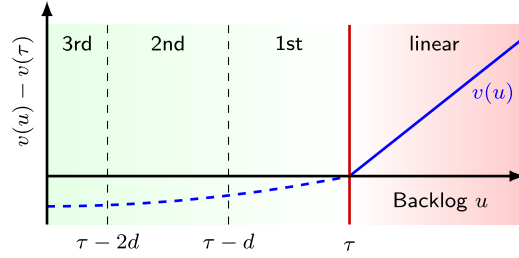


Fig. 1. Value function w.r.t. deadline and $n = 3$ sub-intervals.

3.1. Value function: single job class

In this section, we derive the solution of (5) for $u \leq \tau$. As (1) and (2) give the mean cost rate r for M/D/1, the value function for the tail $u \geq \tau$ is

$$v_0(u) = \lambda \left(\sum_{i=0}^{\lfloor \tau/d \rfloor} \frac{(\lambda(id - \tau))^i}{i!} e^{-\lambda(id-\tau)} \right) (u - \tau).$$

Besides, we know from [3] that (5) expresses $v'(u)$ as a function of $v(u + t)$ with $t \geq 0$. Since $v(u)$ is known for $u \geq \tau$, $v(u)$ can be solved backwards starting from $u = \tau$.

First, define $n \triangleq \lceil \tau/d \rceil$, i.e. n is the finite positive integer number corresponding to the number of intervals with length d that fit below the threshold τ . In other words, n is the number of jobs required to “fill” an empty system to the level where a deadline violation would occur if a new job arrives. We consider sub-intervals of the form

$$\begin{aligned} \mathcal{I}_0 &= (\tau, +\infty), \\ \mathcal{I}_i &= (\tau - id, \tau - (i - 1)d), \quad i = 1, \dots, n. \end{aligned}$$

See Fig. 1. According to (5), $v(u)$ in \mathcal{I}_i for $i = 1, \dots, n$ depends only on $v(u)$ in \mathcal{I}_{i-1} . For $i = 0, \dots, n$, let $\tilde{v}_i(u)$ denote the function equal to the value function in the i th interval, i.e., $\tilde{v}_i(u) = v(u) - v(\tau)$ for $u \in \mathcal{I}_i$, and satisfying, for $u \geq 0$ (cf. analytic continuation)

$$\begin{aligned} \tilde{v}_0(u) &= v_0(u), \\ \tilde{v}'_i(u) + \lambda \tilde{v}_i(u) &= -r + \lambda \tilde{v}_{i-1}(u + d), \quad i = 1, \dots, n. \end{aligned} \tag{6}$$

The $\tilde{v}_i(u)$ are characteristic functions for the value function of the M/D/1 queue w.r.t. deadline. They can be determined from (6) together with the boundary conditions:

$$\tilde{v}_{i+1}(\tau - id) = \tilde{v}_i(\tau - id), \quad i = 0, \dots, n - 1. \tag{7}$$

Then, we could obtain r from $\tilde{v}'_n(0) = 0$, if we did not already have it from (2).

The summation in (2) suggests considering the differences $\tilde{v}_{i+1}(u) - \tilde{v}_i(u)$. Define $y_0(u) = 0$ and, for $i = 1, \dots, n$, let $y_i(u) = \tilde{v}_i(u) - \tilde{v}_{i-1}(u)$. In view of (6) and (7), we find

$$y'_1(u) + \lambda y_1(u) = -\lambda, \tag{8}$$

$$y'_i(u) + \lambda y_i(u) = \lambda y_{i-1}(u + d), \quad i = 2, \dots, n, \tag{9}$$

$$y_i(\tau - (i - 1)d) = 0, \quad i = 1, \dots, n. \tag{10}$$

The next result derives expressions for the y_i functions.

Lemma 1. The functions y_i satisfy

$$y_{i+1}(u) = -1 + e^{-z_i(u)} \sum_{j=0}^i \frac{z_i(u)^j}{j!}, \quad i = 0, \dots, n - 1, \tag{11}$$

where $z_i(u) = \lambda(id + u - \tau)$.

Proof. Using (9) and (10), we find $y_1(u) = -1 + e^{-\lambda(u-\tau)}$. Thus we already know that (11) holds for $i = 0$.

Since the solution of the homogeneous differential equation $y'_i(u) + \lambda y_i(u) = 0$ is $e^{-\lambda u}$, the solution of (9) takes the form $c_i(u)e^{-\lambda u}$, where c_i is such that

$$c'_i(u)e^{-\lambda u} = \lambda c_{i-1}(u + d)e^{-\lambda(u+d)}, \quad \forall u \geq 0, i = 2, \dots, n, \tag{12}$$

$$0 = c_i(\tau - (i - 1)d), \quad i = 2, \dots, n. \tag{13}$$

It follows from (12) and (13) that, for $2 \leq i \leq n$,

$$\begin{aligned} c_i(u) &= \int_{\tau-(i-1)d}^u e^{\lambda t} (\lambda c_{i-1}(t+d)e^{-\lambda(t+d)}) dt \\ &= \lambda e^{-\lambda d} \int_{\tau-(i-1)d}^u c_{i-1}(t+d) dt. \end{aligned}$$

We proceed by induction. First, we assume that (11) holds for $i = k - 1$, where $1 \leq k \leq n$. Then, for $1 \leq i \leq n - 1$, observe that $z_i(t+d) = z_{i+1}(t)$, and

$$\begin{aligned} \int_{\tau-(i+1)d}^u c_{i+1}(t+d) dt &= \int_{\tau-(i+1)d}^u e^{\lambda(t+d)} \left(-1 + e^{-z_i(t+d)} \sum_{j=0}^i \frac{z_i(t+d)^j}{j!} \right) dt \\ &= -\frac{1}{\lambda} [e^{\lambda t}]_{\tau-id}^{u+d} + e^{-\lambda(id-\tau)} \sum_{j=0}^i \int_{\tau-(i+1)d}^u \frac{z_{i+1}(t)^j}{j!} dt. \\ &= -\frac{1}{\lambda} [e^{\lambda t}]_{\tau-id}^{u+d} + e^{-\lambda(id-\tau)} \sum_{j=0}^i \frac{1}{\lambda} \left[\frac{z_{i+1}(t)^{j+1}}{(j+1)!} \right]_{\tau-(i+1)d}^u \\ &= \frac{1}{\lambda} e^{\lambda(u+d)} \left(-1 + e^{-z_{i+1}(u)} \sum_{l=0}^{i+1} \frac{z_{i+1}(u)^l}{l!} \right). \end{aligned} \tag{14}$$

Then, for $k = 1, \dots, n - 1$,

$$\begin{aligned} c_{k+1}(u) &= \lambda e^{-\lambda d} \int_{\tau-kd}^u c_k(t+d) dt \\ &= e^{\lambda u} \left(\lambda e^{-\lambda(u+d)} \int_{\tau-kd}^u c_k(t+d) dt \right) \\ &\stackrel{(14)}{=} e^{\lambda u} \left[-1 + e^{-z_k(u)} \sum_{l=0}^k \frac{z_k(u)^l}{l!} \right]. \end{aligned} \tag{15}$$

Thus,

$$y_{k+1}(u) = -1 + e^{-z_k(u)} \sum_{l=0}^k \frac{z_k(u)^l}{l!}, \text{ for } k = 1, \dots, n - 1$$

and (11) holds by induction on k . \square

We find an explicit result for the value function.

Theorem 2. The value function for an M/D/1 queue with respect to deadline at time τ with a unit violation cost is

$$v(u) - v(\tau) = v_0(u) - m + \sum_{i=0}^{m-1} \left(e^{-z_i(u)} \sum_{j=0}^i \frac{z_i(u)^j}{j!} \right), \tag{16}$$

where $m = \max \left\{ \left\lceil \frac{\tau-u}{d} \right\rceil, 0 \right\}$ and $z_i(u) = \lambda(id + u - \tau)$.

Proof. Follows directly from Lemma 1. \square

Note that, in accordance with (3), $v(u)$ in (16) reduces to a linear function when $u \geq \tau$ and $m = 0$. Moreover, the latter sum is approximately $e^{z_i(u)}$ when i is large, and thus replacing m with any m^* less than m yields an approximation for $v(u)$. Alternatively, one can also write

$$v(u) = v_0(u) - m + \sum_{i=0}^{m-1} \frac{\Gamma(i+1, id + u - \tau)}{i!}, \tag{17}$$

where $\Gamma(a, z)$ is the incomplete gamma function. Given the value function, we can write down the admission cost (marginal cost) $a(u) = v(u+d) - v(u) + \mathbf{1}(u \geq \tau)$.

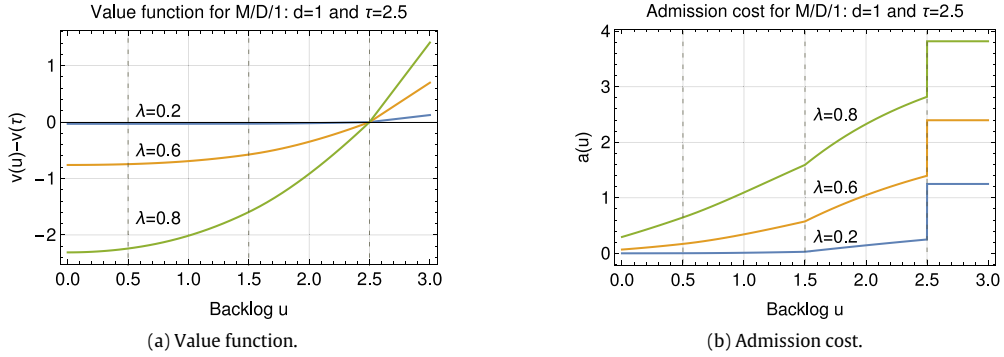


Fig. 2. Value function and the corresponding admission costs for an M/D/1 queue.

Corollary 3. The admission cost of a job with a deadline at time τ to the M/D/1 queue with backlog u is

$$a(u) = 1 + \rho \sum_{i=0}^{\lfloor \tau/d \rfloor} \frac{z_i(0)^i}{i!} e^{-z_i(0)} - \sum_{i=0}^{m-1} \frac{z_i(u)^i}{i!} e^{-z_i(u)}. \tag{18}$$

Note that the first summation in (18) is a constant (cf. the linear term). Recall that $v(u_2) - v(u_1)$ corresponds to the expected difference in the number of deadline violations between a system that has an initial backlog of u_2 and a system that is initially in state u_1 . Similarly, the admission cost $a(u)$ tells us the expected increase in the number of deadline violations if a job is admitted to the system currently in state u , including the cost for the job itself.

In the general case, for the M/G/1 queue with several reasonable cost structures, including deadline violations and latency, it holds that

$$v(0) = -\frac{r}{\lambda} + E[v(X)],$$

where r denotes the corresponding mean cost rate (e.g., $\lambda E[W \geq \tau]$ or $\lambda E[T]$). The above yields a simple identity for the mean admission cost to an empty system,

$$E[a(0)] = \frac{r}{\lambda}.$$

We can verify that this holds also for the M/D/1 queue with the deadline cost structure, i.e., (18) at $u = 0$ reduces to

$$a(0) = P\{W \geq \tau\}, \quad \forall \tau \geq 0.$$

As $v(u)$ for the M/D/1 queue, given in (16), is strictly increasing and convex, $a(u)$ is an increasing function of u . Moreover, for $u < \tau$, $a(u) = v(u + d) - v(u)$, and hence

$$a(u) < v(\tau + d) - v(\tau) = \frac{\rho}{1 - \rho} P\{W < \tau\}.$$

Therefore, the following bounds hold for $a(u)$,

$$P\{W \geq \tau\} \leq a(u) < \frac{\rho}{1 - \rho} P\{W < \tau\}, \quad u < \tau.$$

3.2. Numerical example

Let $d = 1$ and $\tau = 2.5$ so that $n = 3$. The corresponding value function and admission cost are illustrated in Fig. 2 for $\lambda \in \{0.2, 0.6, 0.8\}$. The value function is smooth (except at $u = \tau$), whereas the admission cost behaves quite differently. For example, the unit cost due to the immediate cost of a deadline violation when $u \geq \tau$ shows clearly.

4. M/D/1 with multiple job classes

In this section, we extend the system model and consider the multi-class scenario, where all jobs have the same fixed service time d , but their deadlines and deadline violation costs can vary. More specifically, we assume k job classes such that class i jobs have deadline τ_i (from the arrival time) and each violation for class i jobs costs H_i . The corresponding (Poisson) arrival rates are $\lambda_1, \dots, \lambda_k$, and are such that $\lambda d = \rho < 1$, where $\lambda = \sum_i \lambda_i$ (i.e., a stable system). For convenience, we further define $p_i = \lambda_i/\lambda$.

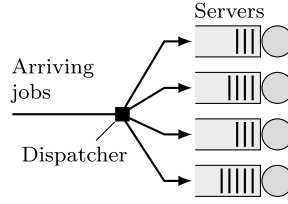


Fig. 3. Dispatching system with $m = 4$ parallel servers.

Let $v_i(u)$ denote the value function of a system with arrival rate λ and deadline τ_i . As $v_i(u) - v_i(0)$ corresponds to how many jobs more on average exceed the deadline τ_i if the initial backlog is u instead of zero, then on average $p_i(v_i(u) - v_i(0))$ of them belong to class i (superposition of Poisson arrival processes), and, as class i violations cost H_i , we have

$$v(u) - v(0) = \sum_i p_i E[H_i](v_i(u) - v_i(0)), \quad (19)$$

where each $v_i(u) - v_i(0)$ is given by (16) with (λ, τ_i) . Note that this is valid because all job classes have the same service time d and are treated the same way under FCFS, and we can also assume that costs are paid upon arrival.

Similarly, the admission cost to the system can be written out, where the immediate cost is included only for the class of the arriving job. That is, if (18) is used, the admission cost of a class j job with violation cost h to an M/D/1 queue in state u is

$$a(u, j) = h \cdot \mathbf{1}(u \geq \tau_j) + \sum_i p_i E[H_i] (a_i(u) - \mathbf{1}(u \geq \tau_i)).$$

Note that h can be replaced with $E[H_j]$ if the violation cost of the given job is unknown.

We note that *without any technical difficulties*, we can extend the model so that each job class can also have several deadlines with arbitrary violation costs. That is, a penalty is paid for each deadline that is violated for the same job. Then we can approximate any cost structure based on the waiting and/or sojourn times. For example, a cost structure for a single class with unit violation costs $h_i = h$ and deadlines $\tau_i = ih$, where $i = 0, 1, \dots$, converges to the cost structure where each job incurs a cost equal to its waiting time when $h \rightarrow 0$. This is equivalent to the (mean) waiting time. However, for clarity of presentation we omit such examples.

5. Parallel servers

In this section, we consider a dispatching system with parallel servers, as illustrated in Fig. 3. First we develop efficient dispatching policies based on the new results given in the previous sections, and study them analytically. Then we evaluate our heuristics numerically through simulations.

5.1. Model and reference policies

We consider the following model for a multi-server system:

- m parallel servers with service times d_i , $i = 1, \dots, m$.
- s job classes with parameters (λ_i, τ_i, h_i) , $i = 1, \dots, s$.

The offered load to the system is $\rho_{\text{tot}} = \sum_i \lambda_i / \sum_i 1/d_i$, which is assumed to be less than one. We consider the following heuristic dispatching policies:

- *Random split* (RND) routes a job to Server i with probability p_i . The splitting probabilities p_i are (typically) chosen so that the offered load ρ_{tot} is balanced among the m servers, $\rho_i = \rho_j$.
- *Class-specific split* (CIQ, class-is-queue) routes class i jobs to server i . Hence, we assume that $m = s$.
- *Least-work-left* (LWL) routes a job to the server with the smallest backlog. Let $\alpha(u_1, \dots, u_m, j)$ denote the chosen server for class j in state u_1, \dots, u_m . Then

$$\alpha_{\text{LWL}}(u_1, \dots, u_m, j) \in \underset{i}{\operatorname{argmin}} u_i.$$

Ties can be resolved in an arbitrary fashion.

- *Dead- k* , introduced in [3], is like LWL, but server $i = 1, \dots, k$ is excluded if $u_i > \tau$. Hence, backlog in the first k servers is rarely above the deadline threshold τ . (Assuming a common deadline, $\tau_i = \tau$, $\forall i$.)

Note that RND and CIQ are *static* policies, i.e., their actions are independent of the system's state. We develop new policies based on the value functions derived in the previous section by carrying out one policy improvement step. More specifically, the standard procedure (see, e.g., [3,5,10,16]) is as follows:

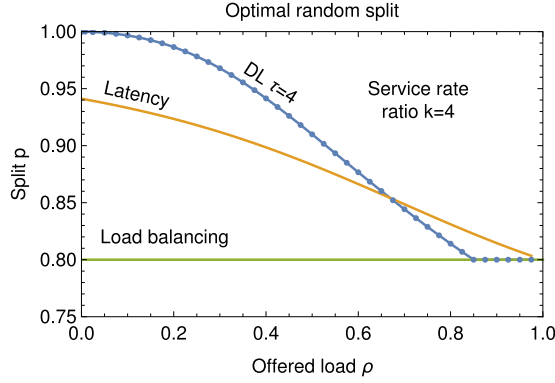


Fig. 4. Static policies in Example #1.

1. Choose a *static* policy α_0 , e.g., RND or CIQ.
2. With α_0 , the system decomposes and each server behaves as an independent M/D/1 queue.
3. Compute the value functions for each M/D/1 queue.
4. Carry out the policy improvement step:

$$\alpha_{\text{FPI}}(u_1, \dots, u_m, j) \in \underset{i}{\operatorname{argmin}} a_i(u_i, j).$$

We refer to these policies with FPI (first-policy-iteration).

5.2. Example #1: Slow server scenario

Let us start with a simple example of a single class and two heterogeneous servers, where Server 1 is faster than Server 2 (cf. slow-server problem). The corresponding processing times are $d_1 = 1$ and $d_2 = 4$ time units, and the target deadline is $\tau = 4$.

The static split is defined by probability p of routing a job to Server 1, so the server-specific arrival rates are λp and $\lambda(1-p)$, respectively. Fig. 4 shows the split probabilities for three static policies: the load balancing split ($p_\rho = 4/5$), the optimal split w.r.t. mean latency (p_W), and the optimal split w.r.t. deadline violations (p_D). For s servers, the optimal split w.r.t. mean latency p_W is obtained by solving (cf. PK mean waiting time formula),

$$\begin{aligned} \min_{p \in P} \quad & \sum_{i=1}^s p_i \frac{\lambda p_i d_i^2}{1 - \lambda p_i d_i} \\ \text{subject to} \quad & \sum_{i=1}^s p_i = 1 \end{aligned} \quad (20)$$

where $p = (p_1, \dots, p_s)$ with $p_i \in P_i = [0, \bar{p}_i]$, d_i denotes the service time of server i , and $\bar{p}_i = \min(1, 1/\lambda d_i) > 0$ ($i = 1, \dots, s$). In our example, $s = 2$ and $d = (1, 4)$. Similarly, the optimal split p_D w.r.t. deadline violations is obtained from

$$\begin{aligned} \min_{p \in P} \quad & \sum_{i=1}^s p_i G(p_i \lambda, d_i) \\ \text{subject to} \quad & \sum_{i=1}^s p_i = 1 \end{aligned} \quad (21)$$

where $G(\lambda, d_i) = P\{W > \tau\}$ and is given in (2). Problems (20) and (21) are separable convex programs which can be solved efficiently in parallel. Methods of solution based on dual decomposition are supplied in Appendix.

Fig. 5 depicts the relative performance of the static policies. The y-axis is the deadline violation probability with the given policy divided by the corresponding probability with the optimal split p_D . We have also included the performance with a single fast server with $d = 4/5$ for comparison.

With dynamic policies the routing decision depends on the state of the system, i.e., the backlogs (u_1, u_2). LWL sends a job to Server 1 if $u_1 \leq u_2$ and otherwise to Server 2. The Dead-1 policy is like LWL, but never sends a job to Server 1 if its backlog is more than τ . Finally, we can also carry out one policy improvement round, e.g., from the load-balancing random split RND; we refer to this policy as FPI.

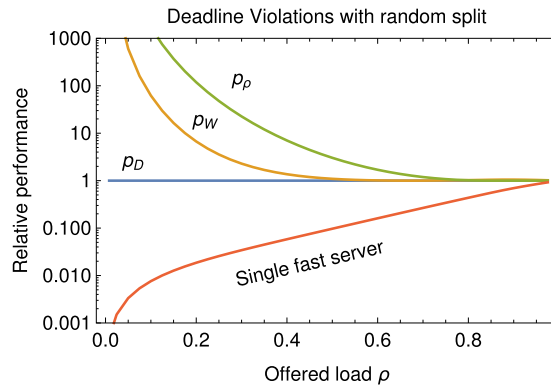


Fig. 5. Performance with static policies.

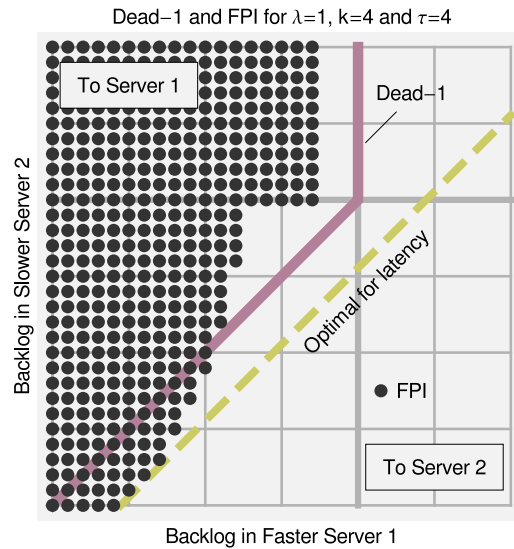


Fig. 6. Dynamic policies Dead-1 and FPI when $\lambda = 1$ and $\rho = 0.8$ in the example scenario #1.

The Dead-1 and FPI policies are illustrated in Fig. 6 for $\lambda = 1$ so that $\rho = 0.8$. The x-axis corresponds to the backlog in Server 1, and y-axis to the backlog in Server 2. Consequently, at the top-left corner, all policies route a job to Server 1, and at the bottom right corner to Server 2. The black dots correspond to routing to Server 1 under FPI, whereas the switch-over curve of Dead-1 is drawn explicitly. For clarity, we have omitted LWL, whose switch-over curve is simply $y = x$. For reference, we have also included the switch-over curve of the optimal dispatching policy when the objective is to minimize the mean latency [17]. We can see that both FPI and Dead-1 “protect” the faster Server 1 from overload. This, however, may cause Server 2 to become unstable when ρ is high, as noted in [3].

Numerical results with dynamic policies LWL, FPI and Dead-1 are depicted in Fig. 7. We have also included the load-balancing RND for reference. It can be seen that initially LWL does a good job, but at higher loads it fails badly. FPI and Dead-1 have the best performance, as expected.

5.3. Example #2: High priority jobs

Suppose next that we have two job classes and two identical servers. The parameters of the job classes are given in Table 1. Here Class 1 corresponds to high priority customers that have both more stringent deadlines and larger deadline violation penalties.

5.3.1. Class-specific heuristics

Here we consider two heuristic static policies: (i) random 50:50 split (RND), and (ii) assigning Class i jobs to Server i (CIQ). Interestingly, both have the same performance, yet they offer two quite different starting points for the policy improvement



Fig. 7. Simulation results with dynamic policies in scenario #1.

Table 1
Job-classes of Example #2.

	Arrival rate	Deadline τ_i	Cost
Class 1	0.5λ	2	10
Class 2	0.5λ	4	1

step. In particular, a deviation from CIQ means that one is determining the states in which one should assign a Class 1 job to Server 2, and vice versa.

Moreover, as the servers are identical, we have one more *trick* at our disposal: at any moment, we can renumber (interchange) the servers and effectively jump to another state.¹ Given we know the value functions, we can choose the renumbering such that the expected costs in future, assuming the given static policy, are minimized. This yields a new policy, FPI-CIQ-S, defined by

$$\operatorname{argmin}_i \left\{ h_j \mathbf{1}(u_i \geq \tau_j) + \min_{\pi} \sum_k v_{\pi_k}(u_k + \mathbf{1}(k=i) \cdot d) \right\},$$

where π iterates over all permutations of $(1, \dots, n)$. The renumbering trick can be applied to any subset of identical servers. Moreover, the resulting dispatching policy is *symmetric*. It is intuitively clear that the optimal dispatching policy must possess the same symmetry.

5.3.2. Resulting policies

The resulting policies are illustrated in Fig. 8 for $\lambda = 1$, i.e., $\rho = 0.5$. The upper row corresponds to the decisions for high priority jobs (class 1), and the lower row corresponds to low priority jobs (class 2). The light red solid lines indicate the threshold τ_i for class i . Below the horizontal line, the backlog in Server 2 is less than τ_i , and to left of the vertical line the backlog in Server 1 is less than τ_i .

In the first column, we have the least-work-left (LWL) policy, which chooses Server 1 when $u_1 < u_2$ (bottom right triangle), and Server 2 when $u_1 > u_2$. Ties, when $u_1 = u_2$, can be resolved in arbitrary fashion in this case.

The second column corresponds to policy iteration when the starting point is (uniform) RND, yielding FPI-RND. In this case, the value functions for both servers are identical, and one obtains LWL (given the ties, indicated with light gray dots, are resolved in the favor of the shorter queue).

The third column has FPI-CIQ, which seems to be quite a sensible policy that “protects” Server 1 in order to minimize the deadline violations of high priority jobs. For example, whenever Server 2 can satisfy a deadline, a job of any class is routed there. Moreover, the low priority jobs can enter Server 1 only if their own server is “full” and Server 1 has no or very little work in the queue.

In the fourth column, we have a scheme combining policy iteration and renumbering, FPI-CIQ+S. This policy is, by construction, symmetric. In our case, the high priority jobs are assigned to the longer queue when both or neither queue can satisfy the target deadline, otherwise the queue that can satisfy the deadline (the shorter queue). Low priority jobs, on the other hand, are typically routed to the longer queue, except when one queue is too long and the other very short or empty. All these properties are quite reasonable.

¹ This is referred to as *rename* in [18] and *switch* in [5].

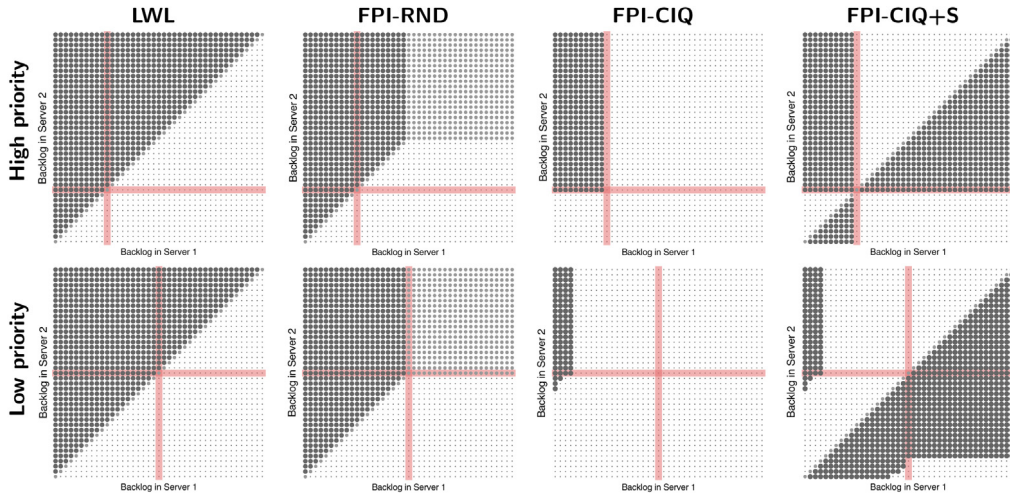


Fig. 8. Routing a job to the “high-priority” Server 1 with FPI based on CIQ and RND. Black dots correspond to Server 1, white dots Server 2, and gray dots mean a tie.

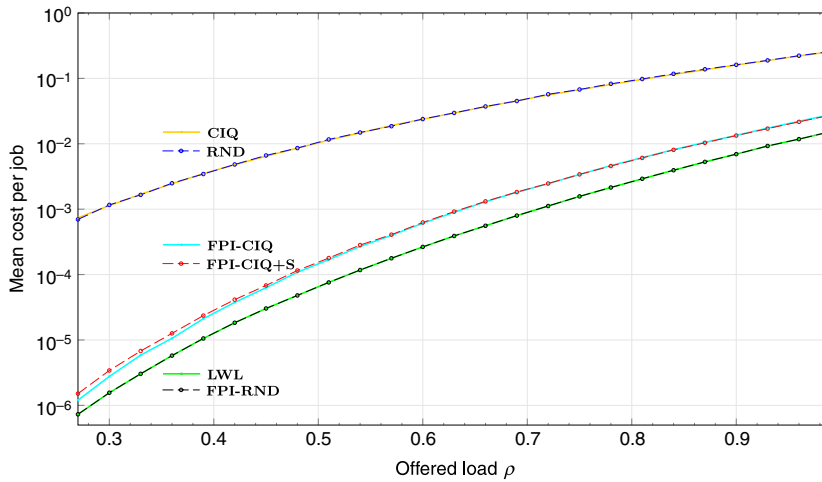


Fig. 9. Simulation results with dynamic policies in scenario #2.

5.3.3. Simulation results

Next we will evaluate the performance of the different dispatching policies. Fig. 9 depicts the simulation results (in log scale on vertical axis) in this rather complex scenario. On the x-axis, we vary the offered load ρ from $\rho = 0.27$ to $\rho = 0.99$. The y-axis corresponds to the average cost per job, i.e., we divide the accumulated costs by the number of jobs. Both RND and CIQ have the same rather poor performance, as expected. First policy iteration on CIQ reduces the deadline violation costs substantially (with or without switch). The best performance for this scenario is achieved after first policy iteration of the RND policy, which behaves like the LWL policy, as predicted by Fig. 8.

6. Summary

Past work has given explicit forms for the value function with respect to the deadline cost structure only for specific cases: (i) in the heavy-traffic regime as $\rho \uparrow 1$, and (ii) when all service times are larger than the (single) deadline. In the heavy-traffic regime, the value function for M/G/1 (with large deadline) is quadratic. When the deadline is smaller than the service time, the value function includes an exponential term.

In this paper, we give the first exact expression for the value function with respect to (possibly multiple) deadlines for a single server queue under arbitrary load. To this end, we assume a Poisson arrival process and a fixed service time, i.e., the M/D/1 queue. The basic result takes the form of a double sum with a finite number of terms. This result is then generalized for the M/D/1 queue with multiple job classes, each having its own target deadline and violation cost. The availability of the

value function enables policy iteration when developing cost-aware dispatching strategies for parallel servers, making these results immediately useful.

Acknowledgment

This work was supported by the Academy of Finland in the FQ4BD project (Grant No. 296206).

Appendix. Optimal static policies

In this appendix we consider the random static policies suggested in Section 5.2 and suggest parallel algorithms for solving (20) and (21).

Minimum latency split. Using dual optimization, Problem (20) can be parallelized and decomposed into s problems of smaller sizes [19–21]. Starting from an arbitrary $y^0 \in \mathbb{R}$, we suggest an iterative optimization algorithm using Newton steps wherever the dual function $g(y)$ is strongly concave, and steepest descent steps elsewhere:

$$y^{k+1} = \begin{cases} \min_i \bar{y}_i + \beta^k \left(\sum_{i=0}^s \bar{p}_i - 1 \right) & \text{if } y^k \leq \min_i \bar{y}_i, \\ \max \left(y^k - \alpha_-^k \frac{g'(y^k)}{\partial_- g'(y^k)}, \min_i \bar{y}_i \right) & \text{if } \min_i \bar{y}_i < y^k < 0, \ g'(y^k) < 0, \\ y^k & \text{if } \min_i \bar{y}_i < y^k < 0, \ g'(y^k) = 0, \\ \min \left(y^k - \alpha_+^k \frac{g'(y^k)}{\partial_+ g'(y^k)}, 0 \right) & \text{if } \min_i \bar{y}_i < y^k < 0, \ g'(y^k) > 0, \\ -\beta^k & \text{if } y^k \geq 0, \end{cases} \quad (\text{A.1})$$

where y is the dual variable, $\bar{y}_i = d_i(1 - (1 - \lambda \bar{p}_i d_i)^{-2})$ with $\bar{y}_i < 0$ if $\lambda d_i < 1$ and $\bar{y}_i = -\infty$ otherwise, g' is the derivative of g , $\partial_- g'$ and $\partial_+ g'$ denote the left and right second derivatives² of g , (β^k) is a static sequence of step sizes,³ and the step sizes (α_{\pm}^k) are determined by an approximate line search routine of the type Armijo [22].

Algorithm (A.1) proves to converge quadratically provided that $\lambda < \sum_{i=1}^s \frac{1}{d_i}$, i.e. as long as the offered load $\rho = \lambda / (\sum_{i=1}^s \frac{1}{d_i})$ is less than 1 [23]. The split policy p_W can then be recovered using

$$x_i^*(y) = \begin{cases} \bar{p}_i & \text{if } y \leq \bar{y}_i, \\ \frac{1}{\lambda d_i} \left(1 - \sqrt{\frac{d_i}{d_i - y}} \right) & \text{if } \bar{y}_i < y < 0, \\ 0 & \text{if } y \geq 0, \end{cases} \quad (\text{A.2})$$

where $(x_1^*(y^k), \dots, x_s^*(y^k)) \rightarrow p_W$. In the particular case when $\rho = 1$, we find the optimal policy $(\frac{1}{\lambda d_1}, \dots, \frac{1}{\lambda d_s})$.

Optimal split with respect to deadline violations. An analogous algorithm – omitted for concision – can be designed for Problem (21) by following a similar procedure. Although the complex structure of G makes it difficult to derive closed-form expressions for the primal minima (x_1^*, \dots, x_s^*) , the latter may be computed numerically using Newton's method. The derivative of the dual function follows from Danskin's theorem [24], i.e. $g'_i(y) = x_i^*(y)$, whereas the second derivative of g can be inferred using the implicit function theorem.

A specificity of (21) is that the problem becomes ill-conditioned as soon as one server offers very short service times relative to the deadline. In that case, the dispatching policy p_D tends to dispatch practically all the jobs to such a server.

² An example of such families of step size sequences satisfies $\beta^k > 0$, $\sum_{k=0}^{\infty} (\beta^k)^2 = 0$, and $\sum_{k=0}^{\infty} \beta^k = \infty$.

³ For this problem the derivatives of the dual function are given, for $y \in \mathbb{R}$, by

$$g'(y) = \sum_{i \in S(y^k)} \frac{1}{\lambda d_i} \left(1 - \sqrt{\frac{d_i}{d_i - y}} \right) + \sum_{i \in T(y^k)} \bar{p}_i - 1,$$

where $S(y) = \{i : \bar{y}_i < y < 0\}$, $T(y) = \{i : y < \bar{y}_i\}$, and

$$\partial_- g'(y) = - \sum_{i \in S(y^k)} \frac{(d_i - y)^{-\frac{3}{2}}}{2\lambda \sqrt{d_i}} - \sum_{i \in \bar{S}(y^k)} \frac{1}{2\lambda d_i^2},$$

$$\partial_+ g'(y) = - \sum_{i \in S(y^k)} \frac{(d_i - y)^{-\frac{3}{2}}}{2\lambda \sqrt{d_i}} - \sum_{i \in \bar{T}(y^k)} \frac{(1 - \lambda \bar{p}_i d_i)^3}{2\lambda d_i^2},$$

where $\bar{S}(y) = \{i : y = 0\}$ and $\bar{T}(y) = \{i : y = \bar{y}_i\}$.

References

- [1] J. Dean, L.A. Barroso, The tail at scale, *Commun. ACM* 56 (2) (2013) 74–80.
- [2] Z. Liu, M.S. Squillante, J.L. Wolf, On maximizing service-level-agreement profits, in: *Proceedings of the 3rd ACM Conference on Electronic Commerce, EC '01*, ACM, New York, NY, USA, 2001, pp. 213–223.
- [3] E. Hyttiä, R. Righter, Routing jobs with deadlines to heterogeneous parallel servers, *Oper. Res. Lett.* 44 (4) (2016) 507–513.
- [4] E. Hyttiä, A. Penttinen, S. Aalto, J. Virtamo, Dispatching problem with fixed size jobs and processor sharing discipline, in: *23rd International Teletraffic Congress, ITC'23*, San Francisco, USA, 2011, pp. 190–197.
- [5] E. Hyttiä, A. Penttinen, S. Aalto, Size- and state-aware dispatching problem with queue-specific job sizes, *European J. Oper. Res.* 217 (2) (2012) 357–370.
- [6] K.R. Krishnan, Joining the right queue: a state-dependent decision rule, *IEEE Trans. Automat. Control* 35 (1) (1990) 104–108.
- [7] E. Hyttiä, R. Righter, S. Aalto, Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure, *Perform. Eval.* 75–76 (0) (2014) 17–35.
- [8] A. Penttinen, E. Hyttiä, S. Aalto, Energy-aware dispatching in parallel queues with on-off energy consumption, in: *30th IEEE International Performance Computing and Communications Conference, IPCCC*, Orlando, FL, USA, 2011.
- [9] E. Hyttiä, J. Virtamo, S. Aalto, A. Penttinen, M/M/1-PS queue and size-aware task assignment, *Perform. Eval.* 68 (11) (2011) 1136–1148.
- [10] K.R. Krishnan, T.J. Ott, State-dependent routing for telephone traffic: Theory and results, in: *IEEE Conference on Decision and Control*, vol. 25, 1986, pp. 2124–2128.
- [11] L. Takács, A single-server queue with Poisson input, *Oper. Res.* 10 (3) (1962) 388–394.
- [12] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, Wiley Interscience, 1975.
- [13] J.W. Roberts, J.T. Virtamo, The superposition of periodic cell arrival streams in an ATM multiplexer, *IEEE Trans. Commun.* 39 (2) (1991) 298–303.
- [14] R. Syski, *Introduction to Congestion Theory in Telephone Systems*, North-Holland, 1986.
- [15] J.T. Virtamo, Numerical evaluation of the distribution of unfinished work in an M/D/1 system, *Electron. Lett.* 31 (7) (1995) 531–532.
- [16] H. Wu, K. Wolter, Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric, in: *Proc. of Valuetools'14*, 2014, pp. 90–97.
- [17] E. Hyttiä, Optimal routing of fixed size jobs to two parallel servers, *INFOR Inf. Syst. Oper. Res.* 51 (4) (2013) 215–224.
- [18] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, M.S. Squillante, Analysis of task assignment with cycle stealing under central queue, in: *Proc. of the 23rd International Conference on Distributed Computing Systems, ICDCS*, Washington, DC, USA, 2003, pp. 628–637.
- [19] D. Bertsekas, *Convex Optimization Theory*, Athena Scientific, 2009.
- [20] D. Bertsekas, J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.
- [21] O. Bilenne, *Distributed Methods for Convex Optimisation: Application to Cooperative Wireless Sensor Networks (Ph.D thesis)*, Technische Universität Berlin, Berlin, 2015.
- [22] L. Armijo, Minimization of functions having lipschitz continuous first partial derivatives, *Pacific J. Math.* 16 (1) (1966) 1–3.
- [23] D. Bertsekas, *Nonlinear Programming*, second ed., Athena Scientific, 1999.
- [24] A. Shapiro, D. Dentcheva, A. Ruszczyński, *Lectures on Stochastic Programming: Modeling and Theory*, MPS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics, Philadelphia, 2009.



Esa Hyttiä is an Associate Professor in the Department of Computer Science at the University of Iceland (HÍ). During 2005–2006, he was with the Norwegian University of Science and Technology (NTNU), Norway, 2006–2009 with Telecommunication Research Center Vienna (FTW), Austria, and 2009–2015 with Aalto University, Finland. He received his Dr.Sc. (Tech.) degree in Electrical Engineering from Helsinki University of Technology (TKK) in 2004. His research interests include performance analysis, modelling, design and optimization of computer and communications systems.



Rhonda Righter is a Professor and past Chair of the Department of Industrial Engineering and Operations Research at the University of California, Berkeley. Before joining Berkeley she was a Professor of Operations Management and Information Systems in the Leavey School of Business at Santa Clara University. Her Ph.D. is in Industrial Engineering and Operations Research from UC Berkeley, her BS is in Applied Math and Business from Carnegie Mellon. Her primary research and teaching interests are in the general area of stochastic modeling and optimization, especially as applied to service, manufacturing, computer communication, and cloud computing systems. She is an associate editor for the *Journal of Scheduling*, *Queueing Systems*, and the *INFORMS Service Science Journal*. She formerly served on the editorial boards of *Management Science*, *Operations Research*, and *Operations Research Letters*. She is the past (founding) Chair of the Applied Probability Society of INFORMS.



Olivier Bilenne is a postdoctoral researcher in the Department of Communications and Networking at Aalto University. He holds an electrical engineering degree from the University of Liège, and a Ph.D. in engineering from the Technical University of Berlin. His research interests include, among others, mathematical optimization, stochastic programming, and queueing theory.



Xiaohu Wu received his Ph.D. degree from Telecom ParisTech, France in 2016. Currently, he is postdoctoral researcher at Aalto University, Finland. He has held visiting positions at University of California at Berkeley in 2015 and Singapore University of Technology and Design from 2011 to 2012. His research interests include queuing theory, approximation algorithms, and cloud computing.