Task Assignment in a Heterogeneous Server Farm with Switching Delays and General Energy-Aware Cost Structure

Esa Hyytiä[a,1,*], Rhonda Righter[b], Samuli Aalto[a]

[a]*Department of Communications and Networking, Aalto University, Finland*
[b]*University of California Berkeley, 4141 Etcheverry Hall, Berkeley, California 94720-1777*

Abstract

We consider the task assignment problem to heterogeneous parallel servers with switching delay, where servers can be switched off to save energy. However, switching a server back on involves a constant server-specific delay. We will use one step of policy iteration from a starting policy such as Bernoulli splitting, in order to derive efficient task assignment (dispatching) policies that minimize the long-run average cost. To evaluate our starting policy, we first analyze a single work-conserving M/G/1 queue with a switching delay and derive a value function with respect to a general cost structure. Our costs include energy related switching and processing costs, as well as general performance-related costs, such as costs associated with both means and variability of waiting time and sojourn time. The efficiency of our dispatching policies is illustrated with numerical examples.

Keywords: Task Assignment, M/G/1-Queue, Switching Delay, Energy-aware, MDP

1. Introduction

In task assignment (dispatching) problems, arriving jobs are assigned to available servers immediately upon arrival. In the basic setting, these servers are modelled as work-conserving queues with a constant service rate. Task assignment problems arise, e.g., in transportation (e.g., street toll booths), manufacturing systems, (data) traffic routing, super computing, cloud computing, data centers (e.g., web server farms) and other distributed computing systems. A typical objective is to minimize the mean waiting time or the mean sojourn time (i.e, response time, latency or delay, where the terminology varies with the application). Recently, energy consumption, e.g., in data centers and distributed computing in general, has become an important consideration. Therefore, in this paper we assume that the general cost structure includes energy consumption related costs in addition to the usual performance metrics such as the mean sojourn time. Moreover, we consider a model where a server can be switched off in order to save energy. However, switching a server back on is assumed to take a certain setup time d during which no job can be served, and may incur a cost. The optimal dispatching policy finds an appropriate trade-off between the criteria of minimizing energy costs while maximizing customer satisfaction.

We approach this challenging problem in the framework of Markov decision processes (MDP) [1–3]. First we consider an isolated M/G/1 queue with a switching delay and derive the so-called size-aware value functions with respect to different terms in the cost structure. We consider two types of costs: (i) costs defined by a rate (e.g., energy consumption) and (ii) immediate costs (e.g., switching the server on or off). For some quantities such as the waiting time, it is possible to define the cost either as a cost rate (i.e., the number of waiting jobs) or as an immediate cost (the waiting time becomes known upon an arrival to a FCFS queue). We will utilize both conventions in this paper.

The value functions characterize the difference in the long-term mean costs between initial states. Using the value functions as inputs to the MDP-based heuristics of first policy iteration (FPI) and lookahead, we derive energy-

*Corresponding author

Email addresses: esa.hyytia@aalto.fi (Esa Hyytiä), rrighter@ieor.berkeley.edu (Rhonda Righter), samuli.aalto@aalto.fi (Samuli Aalto)

[1]This research was carried out while Esa Hyytiä was visiting professor Rhonda Righter at UC Berkeley.

and state-aware dispatching policies, which also take into account the switching delays in the servers. The resulting policies are evaluated and illustrated in numerical examples.

In this paper, we extend earlier results [4, 5] for the M/G/1 queue in several respects. First, we include a switching delay, which is an important system characteristic when, e.g., a server in a data center is switched off in order to save energy. Second, we also consider general holding cost functions, and derive explicit expressions for the second moments of the waiting time and sojourn time, which allow one to improve fairness in the system. Moreover, the energy consumption model also includes switching costs and is thus more comprehensive than in [6]. The new results are the *size-aware value functions* for the M/G/1 queues, when the objective is to minimize the (mean) backlogs, waiting times, sojourn times, or any combination of thereof (as well as higher moments). These results also yield the corresponding value functions for a system of parallel servers (our model for a server farm) operating under any static policy, which are prerequisites for the efficient dynamic policies via the FPI and Lookahead approaches.

1.1. Related work

Dispatching problems have been studied extensively in the literature. Within each queue, first-come-first-served (FCFS) scheduling is usually assumed, but other scheduling disciplines have also been studied. Only a few optimality results are known for the dispatching problems, and these generally require homogeneous servers. In general, the optimal decision depends on the available information.

Round-Robin (RR), followed by FCFS, is shown to be the optimal policy when it is only known that the queues were initially in the same state, and the dispatching history is available [7–9].

Crovella et al. [10] and Harchol-Balter et al. [11] assume that the dispatcher is aware of the size of a new job, but not of the state of the FCFS queues, and propose *Size-Interval-Task-Assignment* (SITA) policies, where each queue receives the jobs within a certain size interval (e.g., short jobs to one queue, and the rest to another). Feng et al. [12] later showed that SITA is the optimal size-aware static policy for homogeneous servers.

When the number of tasks per server is available, the intuitive *Join-the-Shortest-Queue* (JSQ) policy assigns a new job to the server with the fewest tasks. Assuming exponentially distributed interarrival times and job sizes, and homogeneous servers, [13] shows that JSQ with FCFS minimizes the mean waiting time. Since then the optimality of JSQ has been shown in many other settings [7, 14–18]. Recently, Akgun et al. [19], have shown the optimality of JSQ for G/M/1 queues under very general assumptions. In contrast, Gupta et. al consider JSQ with the *processor-sharing* (PS) scheduling discipline in [20]. Whitt [21], on the other hand, provides several counterexamples with non-exponential services where the JSQ/FCFS policy is not optimal, even when servers are homogeneous.

If the queue-specific backlogs (unfinished work, workload) are available, then the *Least-Work-Left* (LWL) policy chooses the queue with the smallest backlog, i.e., it is the greedy (myopic) policy minimizing the waiting time of the new job. Interestingly, the M/G/k system with a central queue is equivalent to LWL, which means that at no time instance, a server is idle at the same time when a job is waiting in some queue. Daley [22], based on Foss's work [23], has shown that G/G/k (i.e., LWL with general inter-arrival times) stochastically minimizes both the maximum and total backlog with identical servers at an arbitrary arrival time instance. In contrast, the counterexample given by Stoyan [24] shows that pathwise RR can yield both a lower waiting time and a lower total backlog (at arrival times).

In general, simple policies such as RR, JSQ and LWL will no longer be optimal when there is switching delay, even in the case of homogeneous servers. Using one step of policy iteration (FPI) in the MDP framework is a promising approach to developing good heuristics for the dispatching problems with heterogeneous servers. Krishnan [25] has utilized it for minimizing mean sojourn time with parallel M/M/s-FCFS servers. See also Aalto and Virtamo [26]. Recently, FCFS, LCFS, SPT and SRPT queues were analyzed in [4, 5] with a general service time distribution. Similarly, PS is considered in [27, 28]. The key idea with the above work is to start with an arbitrary static policy, and then carry out the first policy iteration (FPI) step utilizing the value functions (relative values of states). This general approach is due to Norman [29]. See also [30, 31] for M/G/1 and M/Cox(r)/1, and [32, 33] for blocking systems.

Server farms with switching delays have only been considered recently, and only for homogeneous servers. Artalejo et. al [34] give steady-state results for an M/M/k with setup delays, where idle servers are switched off and at most one server can be in an exponentially distributed setup phase. Gandhi and Harchol-Balter [35] consider an M/G/k with an exponential setup delay and give an exact formula for the mean sojourn time in an M/M/k also assuming that at most one server can be undergoing the switching on process. In [36, 37], Gandhi et. al consider an M/M/k with three different power saving policies: (i) keep idle servers running, (ii) switch idle servers on/off as needed and (iii)

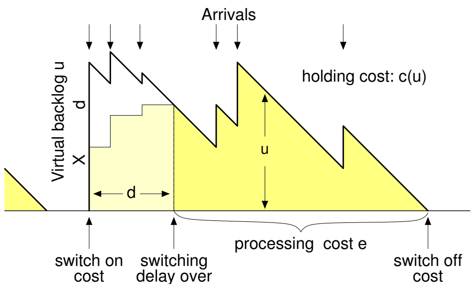


Figure 1: Cost structure illustrated for a single server.

switch idle servers off with at most one server undergoing the switching on (as in [34, 35]). Similarly, Mitrani [38] also considered a scheduling problem with a central queue. In contrast, Maccio and Down [39] considered different switching on/off extensions to an M/G/1 and then applied random routing in the multi-server context. Finally, Wierman et. al [40] consider a capacity management problem in a longer time scale, for which they give rules based on the variations in the arrival rate.

In contrast, switching off a server is understood much better in the context of single-server queues. In addition to switching delay *vacation models* and *removable servers* have been used to describe systems where the server is not necessarily readily available. Welch [41] considers an M/G/1 queue in which the service time of the first customer (a customer arriving to an empty system) obeys a different distribution than the service time of the other customers (customers arriving to a busy system). The Laplace-Stieltjes transform (LST) is derived for the waiting and sojourn times. As a special case, one obtains the M/G/1 queue with switching on delay. We note that it is straightforward to generalize our results on the different value functions to this more general setting. See also Bischof [42] for an analysis of the M/G/1 queue with vacations and setup times. In [43], Hassan and Atiquzzaman also consider an M/G/1 queue with *delayed vacation* and setup time. That is, a server waits a certain time T after the last job has departed before it is switched off. Once switched off, there is a switching on delay C. Explicit expressions for the mean waiting time and the rate of busy cycles are derived.

In the above work, the additional delay is assumed to be a given property of the system, whereas Artalejo [44] considers minimization of different cost functions by delaying the start of service according to N, T and D policies. With these control policies, the server is switched off when the last customer departs. With N policy, the service starts once the number in queue exceeds N [45, 46], and with D policy, when the backlog exceeds D. T policy, on the other hand, activates the server T time units after the end of the last busy period [47]. See also [48, 49] for overviews.

1.2. Outline of the paper

In Section 2, we describe our switching delay model and cost structure, and we give some preliminary results. In Section 3, we analyze an M/G/1 queue with a switching delay and derive value functions with respect to each cost component. We use these results in Section 4 to derive efficient cost-aware dispatching policies, which we evaluate numerically. Section 5 concludes the paper.

2. Preliminaries

2.1. Dispatching system and cost structure

We consider a system of work-conserving parallel queues with a constant *switching-on delay*, or startup time, denoted by d. The switching-off delay is assumed to be sufficiently small so that it can be neglected. New jobs are routed to the available servers immediately upon arrival. There is no jockeying, i.e., assignments are irrevocable. The problem of choosing the queue is known as the dispatching or task assignment problem in the literature.

The queue-specific cost structure is similar to one Feinberg and Kella studied in [50] in the context of a single M/G/1 queue. The per queue cost structure consists of the three components as illustrated in Fig. 1:

(i) switching costs $(k_{\text{on}}, k_{\text{off}})$ (per cycle),

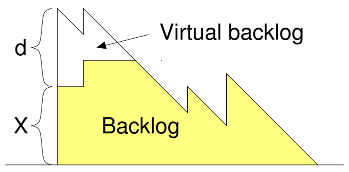


Figure 2: A busy period in M/G/1 with switching delay.

(ii) processing costs $(e_{\text{off}}, e_{\text{start}}, e_{\text{on}})$ (per unit time),

(iii) holding cost $c(u)$ (per unit time), where the virtual backlog (discussed in more detail below) is u.

The *switching costs* are associated with state transitions when a server is switched on or off: k_{on} denotes the cost when a job arrives to an empty system, and k_{off} the cost when the server becomes idle and it is switched off. Without loss of generality, because we are interested in long-run average costs, we can assume that $k_{\text{off}} = 0$ and $k_{\text{on}} = k$.

The *processing costs* are defined by a triple $(e_{\text{off}}, e_{\text{start}}, e_{\text{on}})$: when the server is idle the cost rate is e_{off}, during the startup time the cost rate is e_{start}, and otherwise it is e_{on}. However, without loss of generality, we can add the costs incurred during the startup period, $d \cdot e_{\text{start}}$, to the switching cost k, and further assume $e_{\text{on}} = e$ and $e_{\text{off}} = 0$.

The *running costs* are an important special case when a server incurs costs at the fixed rate of e whenever it is not switched off (i.e., $e_{\text{start}} = e_{\text{on}} = e$). This is obtained with the switching cost of $k = de$.

Finally, the *holding cost* rate is some non-negative and increasing function of the current backlog (workload). We identify two types of backlog:

(i) the actual backlog u^a corresponding to the sum of the (remaining) service times, and

(ii) the virtual backlog u which includes the possible remaining switching delay.

This is illustrated in Fig. 2. Clearly, $u^a \le u$ and when the switching delay $d \to 0$, the two become equal, $u^a = u$. Moreover, $u > 0 \Leftrightarrow u^a > 0$ as we only start the server when work arrives. The actual backlog u^a may be important, e.g., when dimensioning buffers, while the virtual backlog u represents the waiting time of an arriving customer under FCFS scheduling (from PASTA). Therefore, in this paper we focus on the latter and use the term *backlog* to refer to the virtual backlog. The corresponding holding cost rate is denoted by $c(u)$.

The processing, running and switching costs represent operating costs while the holding cost is typically associated with the quality of service. We note that this cost structure is *insensitive to the scheduling discipline* within the class of work-conserving scheduling disciplines, although the interpretation of backlog as waiting time requires FCFS. Similar cost structures have been considered in the context of the T-, $N-$ and D-policies for a single M/G/1-queue with a removable server; see, e.g., [44, 51].

2.2. *M/G/1 with a switching delay and busy period*

Instead of analyzing the complete system of parallel queues with an arbitrary dispatching policy, we assume a static (basic) policy, which allows us to analyze the queues independently. In particular, in this case the behavior of each queue is that of an M/G/1 queue (with a switching delay).

We let λ denote the arrival rate to an M/G/1 queue and $X_i \sim X$ denote the i.i.d. services times. Thus, the offered load is $\rho = \lambda\,\mathrm{E}[X]$. The conditional mean remaining busy period $b(x)$ for a work-conserving M/G/1 queue with an initial backlog of x is [52]

$$b(x) = \frac{x}{1-\rho}. \tag{1}$$

As illustrated in Fig. 2, a busy period with a switching delay starts with a (virtual) backlog equal to $X + d$, where X denotes the job size and d is the switching delay. Thus the mean busy period is

$$\mathrm{E}[B] = \frac{d + \mathrm{E}[X]}{1-\rho}.$$

Each busy cycle is stochastically independent and identical, and we have

$$\mathrm{P}\{\text{system busy}\} = \frac{\mathrm{E}[B]}{1/\lambda + \mathrm{E}[B]} = \frac{\rho + \lambda d}{1 + \lambda d}, \quad \text{and} \quad \mathrm{P}\{\text{system empty}\} = \frac{1-\rho}{1+\lambda d}. \tag{2}$$

The mean length of a busy-idle cycle is

$$\mathrm{E}[T_{\text{cycle}}] = \frac{1}{\lambda} + \mathrm{E}[B] = \frac{1 + \lambda d}{\lambda(1-\rho)}. \tag{3}$$

2.3. Value functions and admission costs

The central notion in this paper is the value function $v(u)$,

$$v(u) \triangleq \lim_{t\to\infty} \mathrm{E}[V_u(t) - r \cdot t], \tag{4}$$

where $V_u(t)$ denotes the costs incurred during $(0, t)$ when initially in state u and r is the mean cost rate. We have implicitly assumed a stable and ergodic system ($\lambda\,\mathrm{E}[X] < 1$). The important quantity for using the value function to find good dispatching policies is the difference,

$$v(u') - v(u),$$

which describes how much more or less it costs to start a system from state u' instead of state u. Therefore, any constant term in $v(u)$ is unimportant. Each new job admitted to an M/G/1 queue tends to increase total costs. Letting u denote the current state and u' the state after an arrival of a size x job, we have

$$u' = \begin{cases} u + x, & \text{if } u > 0, \\ d + x, & \text{if } u = 0. \end{cases}$$

where the latter case includes the switching delay due to starting the server. The marginal *admission cost* of a new job with size x to a queue with backlog u is

$$a(u, x) \triangleq r_\mathrm{a} + v(u') - v(u),$$

where r_a is the *immediate cost* due to the arrival, and the value function $v(u)$ deals with the future costs. Both r_a and $v(u)$ depend on the particular cost structure. The processing and holding costs are per unit time and there is no immediate cost. In contrast, when a new arrival triggers a switch on, the immediate cost is $r_\mathrm{a} = k$.

3. Cost Analysis

In this section, we analyze a single M/G/1 queue with a switching delay. In particular, we derive value functions with respect to the different components in the cost structure. The main results are the new *size-aware value functions* for M/G/1 with switching delay (Propositions 1-7) that enable the improvement of any static dispatching policy, yielding efficient cost-, energy- and state-aware dispatching policies for server farms (see Section 4).

3.1. Operating costs

We start with analysing the switching and processing costs, which together comprise the operating cost. These costs depend only on whether the server is running or not, irrespective of the order in which the jobs are served. Consequently, these results are generic in the sense that they hold for any work conserving scheduling discipline.

Consider first the **switching costs**. The mean switching cost rate r_S is the switching cost k divided by the mean cycle length given by (3),

$$r_S = \frac{\lambda(1-\rho)}{1+\lambda d} \cdot k. \tag{5}$$

We observe that the mean cost rate is insensitive to the job size distribution, and that the switching delay d decreases the mean switching cost rate by a factor of $1/(1 + \lambda d)$.

Proposition 1. *The value function with respect to the switching costs is*

$$v_S(u) - v_S(0) = -\frac{\lambda u}{1 + \lambda d} \cdot k. \tag{6}$$

Proof The value of state u is

$$v_S(u) = \frac{u}{1-\rho}(0 - r_S) + v_S(0),$$

where the first term corresponds to time interval just before the system becomes empty (on average $u/(1-\rho)$) during which no switching costs are incurred. Then $v_S(0)$ takes care of the future. ■

Therefore, the admission cost of a new job with size x to an M/G/1 queue in terms of switching costs is

$$a_S(u, x) = 1(u = 0) \cdot k + v_S(u') - v_S(u),$$

where $u' = u + x + 1(u = 0)\, d$, giving

$$a_S(u, x) = \frac{1(u = 0) - \lambda x}{1 + \lambda d} \cdot k.$$

Processing costs are accrued at rate e (per unit time) whenever the server is serving a job. The mean processing cost rate is the fraction of time the server is busy serving jobs, ρ, multiplied by the running cost rate e,

$$r_P = \rho \cdot e. \tag{7}$$

Proposition 2. *The value function with respect to the processing costs is*

$$v_P(u) - v_P(0) = u \cdot e. \tag{8}$$

Proof The proof is similar to the switching cost case (Proposition 1) and omitted for brevity. ■

Consequently, the processing cost associated with admission of a new job with size x to an M/G/1 queue is

$$a_P(u, x) = x \cdot e.$$

We note that the value functions of the processing and switching costs are linear functions of the virtual backlog u and *independent* of the job size distribution, while the mean cost rates depend on the mean job size E[X] through ρ.

Running costs are the important special case when a system incurs costs at a fixed rate of e whenever it is not switched off. This is equal to the processing cost rate e and the switching cost of $k = de$, yielding the mean cost rate

$$r_R = \frac{\rho + \lambda d}{1 + \lambda d} \cdot e. \tag{9}$$

Similarly, the value function follows from (6) and (8),

$$v_R(u) - v_R(0) = \frac{u}{1 + \lambda d} \cdot e, \tag{10}$$

and the admission cost of a job with size x is,

$$a_R(u, x) = \frac{1(u = 0)d + x}{1 + \lambda d} \cdot e.$$

Without switching delay, i.e., when $d = 0$, the processing costs are equal to running costs.

3.2. *Holding costs*

Consider next an arbitrary backlog-based holding cost $c(u)$. We assume that both $c(u)$ and the corresponding holding cost value function $v_H(u)$ are continuous and differentiable for $u > 0$. Consequently, for a non-empty state $u > 0$ we can write

$$v_H(u) = (c(u)-r_H)\delta + (1-\lambda\delta)v_H(u-\delta) + \lambda\delta \cdot \mathrm{E}[v_H(u-\delta+X)],$$

where δ is a differential time step. Hence,

$$v'_H(u) = c(u) - r_H + \lambda\,\mathrm{E}[v_H(u+X) - v_H(u)],$$

i.e.,

$$c(u) - r_H = v'_H(u) - \lambda\,\mathrm{E}[v_H(u+X) - v_H(u)]. \tag{11}$$

The latter form is interesting as it allows us to *work out the value function backwards*: choose an arbitrary $v_H(u)$ and (11) gives the corresponding holding cost $c(u)$. In particular, we observe that for a polynomial cost function of degree k, (11) is satisfied by a degree $k+1$ polynomial $v_H(u)$.

Let $v_H(u)$ and $\tilde{v}_H(u)$ denote the value functions with and without a switching delay, respectively, for an arbitrary holding cost $c(u)$. The following elementary relationship holds between these values, where r_H and $\tilde{r}_H$ denote the mean holding cost rate with and without a switching delay of d:

Proposition 3.

$$v_H(u) - v_H(0) = \tilde{v}_H(u) - \tilde{v}_H(0) + \frac{u}{1-\rho} \cdot (\tilde{r}_H - r_H), \tag{12}$$

Proof Without a switching delay,

$$\tilde{v}_H(u) - \tilde{v}_H(0) = h(u) - \frac{u}{1-\rho} \cdot \tilde{r}_H, \tag{13}$$

where $h(u)$ denotes the mean cost accrued during the remaining busy period. For a system with switching delay,

$$v_H(u) - v_H(0) = h(u) - \frac{u}{1-\rho} \cdot r_H, \tag{14}$$

because $h(u)$ is independent of the switching delay, and combining (13) and (14) gives (12). ■

Hence, *the switching delay induces a linear term in the value function.* Next we will utilize (12) to give explicit expressions for value functions w.r.t. backlog based holding costs. These results generalize the corresponding results in [4], which will become clear later in Section 3.3 where the relationship between the holding costs and the waiting time (and sojourn time) will be established.

Constant cost rate: Let us use the subscript $H0$ to denote the constant cost rate (holding costs do not depend on backlog). The constant cost rate is defined as

$$c(u) \triangleq \begin{cases} 1, & \text{for } u > 0, \\ 0, & \text{for } u = 0, \end{cases}$$

which is equivalent to the definition of the running costs (when multiplied by e). The running cost corresponds to a simple ON/OFF energy model and the mean cost rate r_{H0} and the value function $v_{H0}(u)$ are already given in (9) and (10) (applied with $e = 1$). The obtained results naturally satisfy (12). We refer the interested reader to [6] for more details and an alternative proof for the value function in the special case of no switching delay.

Linear cost rate: For linear cost, $c(u) = u$, the mean holding cost rate is

$$r_{H1} = \frac{\lambda\,\mathrm{E}[X^2]}{2(1-\rho)} + \frac{d(2\rho+\lambda d)}{2(1+\lambda d)}. \tag{15}$$

See, e.g., [53], where (8.3.64) on page 419 gives the mean waiting time in an M/G/1-FCFS queue with a random switching delay, from which (15) easily follows. Note that the first term in (15) is $\mathrm{E}[W]$ when $d = 0$, i.e., the Pollaczek-Khinchine formula, and the latter term represents the additional penalty due to the switching delay,

$$r_{H1} - \tilde{r}_{H1} = \frac{d(2\rho+\lambda d)}{2(1+\lambda d)}, \tag{16}$$

which is insensitive to job size distribution.

Table 1: Value functions for holding cost.

$c(u)$	Value function, $v(u) - v(0)$
$1(u{>}0)$	$v_{H0}(u) - v_{H0}(0) = \dfrac{u}{1 + \lambda d}$
u	$v_{H1}(u) - v_{H1}(0) = \dfrac{u^2}{2(1-\rho)} - \dfrac{d(2\rho + \lambda d)u}{2(1-\rho)(1+\lambda d)}$
u^2	$v_{H2}(u) - v_{H2}(0) = \dfrac{1}{3(1-\rho)}u^3 + \dfrac{\lambda\,\mathrm{E}[X^2]}{2(1-\rho)^2}u^2 - \left(\dfrac{3\rho + \lambda d}{3(1-\rho)(1+\lambda d)}d^2 + \dfrac{\lambda(2+\lambda d)\,\mathrm{E}[X^2]}{2(1-\rho)^2(1+\lambda d)}d\right)u$

Proposition 4. *The value function w.r.t. linear holding cost $c(u) = u$ for an arbitrary d is*

$$v_{H1}(u) - v_{H1}(0) = \frac{u^2}{2(1-\rho)} - \frac{ud(2\rho + \lambda d)}{2(1-\rho)(1+\lambda d)}. \tag{17}$$

Proof The value function without the switching delay ($d = 0$) is a quadratic function of u [4],

$$\tilde{v}_{H1}(u) - \tilde{v}_{H1}(0) = \frac{u^2}{2(1-\rho)}, \tag{18}$$

Substituting (16) and (18) into (12) completes the proof. ■

Quadratic cost rate: Consider next the quadratic holding cost, $c(u) = u^2$. The results are similar to those with linear holding cost, but involve higher moments of the service time distribution. The mean holding cost rate is

$$r_{H2} = \frac{3\rho + \lambda d}{3(1+\lambda d)}d^2 + \frac{\lambda(2+\lambda d)\,\mathrm{E}[X^2]}{2(1-\rho)(1+\lambda d)}d + \frac{3\lambda^2\,\mathrm{E}[X^2]^2 + 2\lambda(1-\rho)\,\mathrm{E}[X^3]}{6(1-\rho)^2}. \tag{19}$$

Also the proof is similar and omitted for brevity. We note that the last term in (19) is independent of d, and in particular, the increase in the mean holding cost due to the switching delay d is

$$r_{H2} - \tilde{r}_{H2} = \frac{3\rho + \lambda d}{3(1+\lambda d)}d^2 + \frac{\lambda(2+\lambda d)\,\mathrm{E}[X^2]}{2(1-\rho)(1+\lambda d)}d. \tag{20}$$

Lemma 1. *The value function w.r.t. quadratic holding cost $c(u) = u^2$ without switching delay ($d = 0$) is*

$$\tilde{v}_{H2}(u) - \tilde{v}_{H2}(0) = \frac{1}{3(1-\rho)}u^3 + \frac{\lambda\,\mathrm{E}[X^2]}{2(1-\rho)^2}u^2. \tag{21}$$

Proof Substitute a trial $v(u) = au^3 + bu^2$ into (11). ■

Proposition 5. *The value function w.r.t. quadratic holding cost $c(u) = u^2$ for an arbitrary d is*

$$v_{H2}(u) - v_{H2}(0) = \frac{1}{3(1-\rho)}u^3 + \frac{\lambda\,\mathrm{E}[X^2]}{2(1-\rho)^2}u^2 - \left(\frac{3\rho + \lambda d}{3(1-\rho)(1+\lambda d)}d^2 + \frac{\lambda(2+\lambda d)\,\mathrm{E}[X^2]}{2(1-\rho)^2(1+\lambda d)}d\right)u. \tag{22}$$

Proof Substitute (20) and (21) into (12). ■

Repeating the same procedure, both the mean cost rate and value function can be obtained for an arbitrary holding cost $c(u) = u^k$. The value function depends only on the first k moments of the service time distribution, while the mean holding cost rate depends on the first $(k + 1)$ moments. Thus one can approximate any $c(u)$ with arbitrarily high precision using the Taylor series. The results for $k = 0, 1, 2$ are summarized in Table 1.

3.3. *Waiting time and sojourn time*

A backlog-based holding cost rate $c(u)$ can be seen as a penalty for a large amount of unfinished work. However, often one is interested in *waiting time* or *sojourn time*. A non-empty backlog in an M/G/1-FCFS queue corresponds to the virtual waiting time, $W = U$, as the switching delay and the actual workload are no different from an arriving job's point of view. When a job arrives to an empty system, the initial switching delay must be taken into account and $W = d$. The sojourn time is $T = W + X$, where W and X are independent (with FCFS).

In particular, in addition to the servers incurring costs, one can assume that Job j also incurs costs at a job-specific holding cost rate $\omega_j(t)$, e.g., during its waiting time, so that the total cost Job j incurs is

$$\Omega_j = \int_0^{W_j} \omega_j(t)\, dt,$$

where W_j denotes the (final) waiting time of Job j. With a constant $\omega_j(t) = 1$ one obtains $\Omega_j = W_j$, whereas $\omega_j(t) = t/2$ could correspond to impatient jobs and gives $\Omega_j = W_j^2$, etc. Additionally, $\omega_j(t)$ can be a job-specific random function, e.g., $\omega_j(t) = A_j + B_j t$, where $A_j \sim A$ and $B_j \sim B$ are i.i.d. random variables. In other words, our cost structure generalizes to a wide range of job- and server-specific cost rates (see Appendix B for more details). In this paper, however, we limit ourselves to waiting and sojourn time, i.e., we assume that all jobs are equally important.

3.3.1. *Waiting time W*

Let us consider first the waiting time W. We define the mean cost rate with respect to waiting time as

$$r_W = \lambda \cdot \mathrm{E}[W],$$

i.e., the rate at which the system incurs waiting time. Using (15) and (2) we have

$$r_W = \lambda(r_{H1} + \mathrm{P}\{\text{empty}\} \cdot d) = \frac{\lambda^2\, \mathrm{E}[X^2]}{2(1-\rho)} + \frac{\lambda d(2+\lambda d)}{2+2\lambda d}. \tag{23}$$

Consider next a busy period which starts with an initial backlog of u. Let B_u denote the remaining length of the busy period and N_u the number of new jobs arriving during it. For future arrivals, until the end of the busy period, one can associate the costs in two equivalent ways:

$$\begin{aligned} c_1 &\triangleq \mathrm{E}[c(W_1) + \ldots + c(W_{N_u})], \\ c_2 &\triangleq \lambda\, \mathrm{E}[\int_0^{B_u} c(U_t)\, dt], \end{aligned} \tag{24}$$

where, due to the PASTA property, $c_1 = c_2$ (see Appendix). With $c(u) = u$, the first corresponds to the incurred waiting time, and the latter to the linear holding cost (multiplied with λ).

Proposition 6. *The value function w.r.t. waiting time is*

$$v_W(u) - v_W(0) = \frac{\lambda}{2(1-\rho)} \left(u^2 - \frac{d(2+\lambda d)u}{1+\lambda d} \right). \tag{25}$$

Proof Considering the remaining busy period we have

$$v_W(u) = \mathrm{E}[W_1 + \ldots + W_{N_u}] - r_W\, \mathrm{E}[B_u] + v_W(0),$$

and substituting $r_W = \lambda(r_{H1} + \mathrm{P}\{\text{empty}\} \cdot d)$ gives

$$v_W(u) - v_W(0) = \lambda \left(\mathrm{E}[\int_0^{B_u} U_t - r_{H1}\, dt] - \frac{ud}{1+\lambda d} \right),$$

where the expectation is equal to $v_{H1}(u) - v_{H1}(0)$, yielding (25) by (17). ■

Moreover, the linear holding cost structure neglects the current (admitted) job, for which the immediate cost is

$$r_\text{a} = u + 1(u=0)\cdot d = \begin{cases} u, & u>0, \\ d, & u=0, \end{cases}$$

i.e., $r_\text{a} = u' - x$, where u' is the backlog after the new job has been included. Consequently, the admission cost in terms of the waiting time is $a_W(u,x) = u' - x + v_W(u') - v_W(u)$, giving

$$a_W(u,x) = \begin{cases} u + \dfrac{\lambda x}{2(1-\rho)}\left(2u+x-\dfrac{d(2+\lambda d)}{1+\lambda d}\right), & \text{if } u>0, \\ d + \dfrac{\lambda(d+x)}{2(1-\rho)}\left(x - \dfrac{d}{1+\lambda d}\right), & \text{if } u=0. \end{cases} \tag{26}$$

For $u > 0$, the switching delay translates to a *"discount term"* in the admission cost (the new task pushes the next switching delay further into the future). For $u = 0$, an interesting peculiarity is that accepting a very short job may have a negative admission cost as it saves later arriving jobs from the full switching delay.

Similarly, one may be interested in costs in terms of W^2, for which the mean cost rate is

$$\begin{aligned} r_{W2} &= \lambda\left(r_{H2} + \text{P}\{\text{empty}\}\cdot d^2\right) \\ &= \frac{3+\lambda d}{3(1+\lambda d)}d^2 + \frac{\lambda(2+\lambda d)\,\text{E}[X^2]}{2(1-\rho)(1+\lambda d)}d + \frac{3\lambda^2\,\text{E}[X^2]^2 + 2\lambda(1-\rho)\,\text{E}[X^3]}{6(1-\rho)^2}. \end{aligned} \tag{27}$$

The immediate cost is the job's waiting time squared, $r_\text{a} = (u'-x)^2$. Also the corresponding size-aware value function characterizing the cost for the future arrivals can be determined:

Proposition 7. *The value function w.r.t. squared waiting time W^2 is*

$$v_{W2}(u) - v_{W2}(0) = \frac{\lambda u}{1-\rho}\left(\frac{1}{3}u^2 + \frac{\lambda\,\text{E}[X^2]}{2(1-\rho)}u - \frac{(3+\lambda d)}{3(1+\lambda d)}d^2 - \frac{\lambda\,\text{E}[X^2](2+\lambda d)}{2(1-\rho)(1+\lambda d)}d\right). \tag{28}$$

Proof For the value function w.r.t. the squared waiting time of the future arrivals it holds that

$$v_{W2}(u) = \text{E}[W_1^2 + \ldots W_{N_u}^2] - r_{W2}\,\text{E}[B_u] + v_{W2}(0).$$

Then, with the aid of the equivalence (24), we have

$$v_{W2}(u) - v_{W2}(0) = \lambda\left(\text{E}[\int_0^{B_u} U_t^2 - r_{H2}\,dt] - \frac{d^2 u}{1+\lambda d}\right) = \lambda\left(v_{H2}(u) - v_{H2}(0) - \frac{d^2 u}{1+\lambda d}\right),$$

and substituting (22) gives (28). ■

Alternatively, one can write

$$v_{W2}(u) - v_{W2}(0) = \frac{\lambda u}{1-\rho}\left[\frac{1}{3}\left(u^2 - \frac{(3+\lambda d)}{1+\lambda d}d^2\right) + \frac{\lambda\,\text{E}[X^2]}{2(1-\rho)}\left(u - \frac{2+\lambda d}{1+\lambda d}d\right)\right].$$

The admission cost w.r.t. the squared waiting time is

$$a_{W2}(u,x) = (u'-x)^2 + v_{W2}(u') - v_{W2}(u).$$

3.3.2. Sojourn time T

Sojourn time (total time in system, latency), denoted by T, can be treated similarly. When one can route a given job to Queue 1 or Queue 2, the sojourn time may be a more meaningful quantity than the waiting time, as it also takes

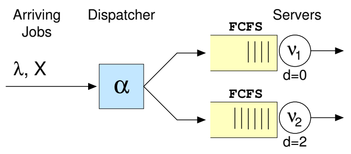


Figure 3: Example dispatching system.

into account the service times of the particular, possibly heterogeneous, servers. The value function with respect to the sojourn time is, obviously, the same (up to a constant term) as for the waiting time,

$$v_T(u) - v_T(0) = v_W(u) - v_W(0),$$

as $T = W + X$. However, the immediate cost w.r.t. sojourn time is

$$r_\text{a} = \begin{cases} u + x, & u > 0, \\ d + x, & u = 0. \end{cases}$$

i.e., $r_\text{a} = u'$. The admission cost in terms of sojourn time reads

$$a_T(u, x) = u' + v_W(u') - v_W(u),$$

i.e., $a_T(u, x) = a_W(u, x) + x$.

We can also obtain similar value functions for the squared sojourn time, T^2.

$$v_{T2}(u) = \text{E}[\sum_{i=1}^{N}(T_i)^2 - \text{E}[T^2] \cdot B_u] + v_{T2}(0).$$

As $T^2 = U^2 + 2\,UX + X^2$, and $U = W$ with FCFS, we obtain

$$v_{T2}(u) - v_{T2}(0) = \text{E}\left[\sum_{i=1}^{N}(W_i)^2\right] - \text{E}[W^2]\text{E}[B_u] + 2\,\text{E}\left[\sum_{i=1}^{N} W_i X_i\right] - \text{E}[W]\text{E}[X]\text{E}[B_u] + \text{E}\left[\sum_{i=1}^{N}(X_i)^2\right] - \text{E}[X^2]\text{E}[B_u],$$

which can be worked out further the same way as earlier. The immediate cost is $r_\text{a} = (u')^2$, and the admission cost

$$a_{T2}(u, x) = (u')^2 + v_{T2}(u') - v_{T2}(u).$$

4. Task Assignment in a Server Farm With Switching Delays

In this section, we consider a system of m heterogeneous parallel servers with Poisson arrivals, i.i.d. service requirements, and server-specific running costs e_i, service rates ν_i, and switching delays d_i. We assume that the holding cost rates related to the service performance, i.e., waiting time and sojourn time, are equal for all servers. However, server-specific cost rates can be introduced without any additional complexity if, e.g., waiting in Queue i is more expensive than in Queue j. This problem, illustrated in Fig. 3, is often referred to as the dispatching or the task assignment problem.

We utilize the results of the previous section to derive dynamic energy- and state-aware dispatching policies, which also take into account whether a particular server is switched off (or not) when idle. For the derivation, we apply two approaches: the first policy iteration (FPI) step of the Markov decision processes [1–3] (see Section 4.3), and the lookahead approach introduced in [54] (see Section 4.4). The resulting policies are compared with a set of well-known reference policies.

4.1. *Reference dispatching policies*

The dispatching decision can be totally random or based on some (partial) information about the job and/or the state of the system. In the class-aware setting, the class of a new job and the corresponding arrival rate and job size distribution are available. In the size-aware setting, the exact size of a job is available. Further, the above information may be available only from the new job (state-free), or also for the jobs currently in the system (state-aware).[2] For example, in size- and state-aware system one knows the exact backlog in the queues. In contrast, in the number-aware case, only the number of jobs in the queues is available. In this paper, we limit ourselves to the size-aware setting, but note that the same approach lends itself also to the other scenarios.

The reference policies are as follows:

- *RND* splits the arriving jobs at random among the servers (Bernoulli split). In particular, we assume the splitting probabilities balance the load, $p_i = \nu_i/\nu_{\text{tot}}$.
- *Size-Interval-Task-Assignment with Equal load (SITA-E)* is a size-aware and state-free policy. With two servers, it assigns jobs shorter than threshold h to Queue 1 and the rest to Queue 2. The threshold is chosen so that the offered loads are equal [10, 11].
- *Greedy* is the individually optimal policy, which minimizes the waiting time (Examples 1 and 2) or the sojourn time of the new job (Example 3). It takes the switching delay d and service rates into account. With equally fast servers, Greedy is equal to LWL (with work being the virtual backlog).
- *Myopic* minimizes the immediate costs (not only the waiting or sojourn time) while ignoring future costs (and arrivals). It is optimal by construction when $\lambda \to 0$.

The first two, RND and SITA-E, are *static* policies, i.e., their decision is independent of the state of the queues. Note that we have excluded, e.g., RR and JSQ as they are not particularly suitable to our heterogeneous setting (at least without appropriate modifications).

4.2. *Switching-off policies*

As was done in [36, 37], we consider two server-specific switching-off policies, `NeverOff` and `InstantOff`, where the former keeps the server always running even when it is idle, and the latter switches the server immediately off when it becomes idle. The rationale is as follows: When the server capacity clearly exceeds the demand, switching the idle servers off makes sense. Similarly, under a heavy load, it is advantageous to keep all the servers running non-stop (to avoid switching delay when $d > 0$). In between, we permit a subset of servers to operate under `NeverOff`, while the rest are `InstantOff`. The task of the dispatching policy is to take these different (a priori fixed) operating modes into account.

Under `NeverOff`, Server i incurs running costs with a constant rate of $r_R = e_i$ (instead of (9)), and there are no switching costs or delays (equivalent to setting $d_i = 0$). For example, the Myopic policy with `NeverOff` considers only the waiting time or sojourn time of the new job.

4.3. *FPI based dispatching policies*

Let $\mathbf{z} = (u_1, \ldots, u_m)$ denote the current state of the system, where u_i is the backlog in Queue i. If Server i is `InstantOff`, then $u_i = 0$ implies that the server is off. Otherwise the server is running. We start with an arbitrary static dispatching policy α, so that the arrival process to each queue is a Poisson process, and each queue behaves as an M/G/1 queue. In particular, the value function of the whole system is the sum of the queue-specific value functions,

$$v_{\mathbf{z}} = \sum_{i=1}^{m} v_i(u_i).$$

Therefore, we can carry out the first policy iteration (FPI) step to obtain an efficient and robust *state-dependent* policy, denoted by FPI-α, that also takes into account the particular cost structure. As the system decomposes into separate queues with a static basic policy α, the resulting FPI policy is simply

$$\text{FPI-}\alpha: \qquad \underset{i}{\operatorname{argmin}}\ a_i(u_i, x_i),$$

[2]Note that state-free policies are more suitable for distributed systems, where task assignment occurs in several locations without coordination.

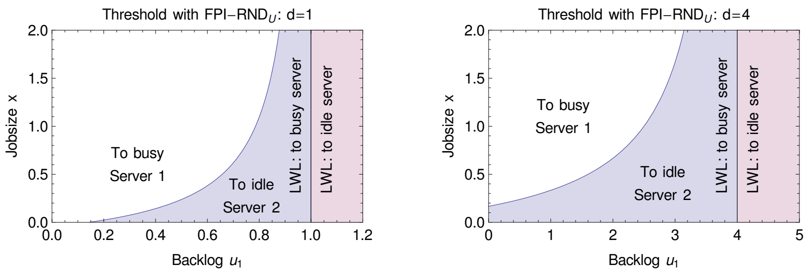


Figure 4: Resulting FPI-RND policies with $\lambda = 0.75$ and $\mathrm{E}[X] = 1$. In the left figure the switching delay is $d = 1$ and in the right $d = 4$.

where $a_i(u, x)$ denotes the admission cost to Queue i as derived in Section 3, and x_i is the service time in Queue i, $x_i = x/\nu_i$, where ν_i is the service rate of Server i. For details, we refer to [4, 5]. The queue-specific value function depends on whether the queue has been designated `NeverOff` or `InstantOff`.

4.3.1. FPI-RND

Let us take a closer look at how FPI-RND works with respect to waiting time. With identical servers and without switching delays, we already know that FPI-RND is equivalent to LWL [4]. Instead, consider otherwise the same setting but with two servers having an identical switching delay of d, and both servers designated as `InstantOff`. In this case, (26) gives that for two non-empty queues with backlogs $u_1 > u_2 > 0$, $a_W(u_1, x) > a_W(u_2, x)$, and, according to the policy improvement principle, the FPI policy reduces again to LWL. However, if, say, Queue 1 is non-empty with a backlog $u_1 > 0$ and Queue 2 is empty, then the question is whether to initiate a switching on process in Queue 2 or not, and here FPI-RND indeed behaves differently from LWL. In particular, FPI-RND assigns a new job with size x to the empty Queue 2 if $a_W(u_1, x) \geq a_W(0, x)$, yielding

$$(d - u_1)(1 - \rho + \lambda x) \leq \frac{\lambda d^2}{2(1 + \lambda d)},$$

where λ and ρ denote the arrival rate and the offered load to each queue (which is the same for every queue under RND). Hence, the decisions of FPI-RND are insensitive to the job size distribution (though it is size-aware), but it is a somewhat more complicated than, e.g., LWL, which chooses the empty Queue 2 if $d < u_1$.

The resulting FPI policy is of the switch-over type and is depicted in Fig. 4 where the queue-specific arrival rate is $\lambda = 0.75$ and the mean job size is $\mathrm{E}[X] = 1$. For comparison, LWL assigns a new job to the idle Queue 2 whenever $u_1 > d$, so the corresponding switch-over curve is a vertical line at $u_1 = d$. We observe that LWL and FPI-RND differ only in the shaded areas at $u_1 < d$, where FPI-RND decides to start the idle server instead of assigning the job to the already busy server. Intuitively, this makes sense if more jobs are anticipated in the near future, which indeed is the case here ($\rho = 0.75$).

4.4. Lookahead based dispatching policies

The FPI based dispatching policies are limited to static basic policies, for which the value function can be determined. The *Lookahead* approach, introduced in [54], tries to get around this shortcoming by taking a "closer look at the future". The basic idea is to consider jointly the new job and the one arriving next. The parameters of the latter job including the time to its arrival, T_1, are unknown. Therefore, one computes the expected cost for actions (i_0, i_1), where i_0 denotes the queue for the current job and i_1 the (tentative) queue for the next. In general, i_1 can also be determined dynamically (e.g., using Greedy or Myopic), but here we limit ourselves to *fixed (tentative) second decisions*. Let us consider the joint optimization of the running costs and the mean sojourn time.

Running costs with `NeverOff` are trivial, and thus we start by considering the running costs with `InstantOff`. Let $(u_1^*, \ldots, u_m^*)$ denote the virtual backlogs after the first assignment according to i_0, i.e., $u_j^* = u_j$ for $j \neq i_0$ and $u_{i_0}^*$ is u_{i_0} with the new job and possible switching delay added. By conditioning on the arrival time $T_1 = t$, the running costs incurred in Queue j until then are

$$\mathrm{E}[\min\{u_j^*, T_1\}]\, e_j = \left(\int_0^{u_j^*} \lambda e^{-\lambda t} t\, dt + e^{-\lambda u_j^*} u_j^*\right) e_j = \frac{1 - e^{-\lambda u_j^*}}{\lambda} e_j.$$

The running costs after time T_1 are given by the corresponding value function (10), where u is now a random variable. Let $U_j(T_1)$ denote the virtual backlog in Queue j *after the assignment of the second job* at time T_1. By conditioning, we have, for $j \neq i_1$,

$$\mathrm{E}[U_j(T_1)] = \int_0^{u_j^*} \lambda e^{-\lambda t}(u_j^* - t)\,dt = u_j^* - \frac{1 - e^{-\lambda u_j^*}}{\lambda},$$

and for Queue i_1, after the assignment of a job with size $X_{i_1} = X/\nu_{i_1}$,

$$\mathrm{E}[U_{i_1}(T_1)] = \mathrm{E}\left[\int_0^{u_{i_1}^*} \lambda e^{-\lambda t}(u_{i_1}^* - t + X_{i_1})\,dt + e^{-\lambda u_i^*}(X_{i_1} + d_{i_1})\right] = d_{i_1} e^{-\lambda u_{i_1}^*} + \frac{\mathrm{E}[X]}{\nu_{i_1}} + u_{i_1}^* - \frac{1 - e^{-\lambda u_{i_1}^*}}{\lambda}. \tag{29}$$

Hence, using (10), the running costs for action (i_0, i_1) are

$$L_R(i_0, i_1) \triangleq \left(d_{i_1} e^{-\lambda u_{i_1}^*} + \frac{\mathrm{E}[X]}{\nu_{i_1}}\right)\frac{e_{i_1}}{1 + \lambda_{i_1} d_{i_1}} + \sum_{j=1}^{m}\left(\frac{\lambda_j d_j (1 - e^{-\lambda u_j^*})}{\lambda} + u_j^*\right)\frac{e_j}{1 + \lambda_j d_j}. \tag{30}$$

When comparing Lookahead actions, we can subtract the same value from all costs without affecting their relative value. Therefore, it is possible to eliminate the summation over m in the above, yielding

$$L_R^*(i_0, i_1) \triangleq \left(d_{i_1} e^{-\lambda u_{i_1}^*} + \frac{\mathrm{E}[X]}{\nu_{i_1}}\right)\frac{e_{i_1}}{1 + \lambda_{i_1} d_{i_1}} + \left(\frac{\lambda_{i_0} d_{i_0}}{\lambda}(e^{-\lambda u_{i_0}} - e^{-\lambda u_{i_0}^*}) + u_{i_0}^* - u_{i_0}\right)\frac{e_{i_0}}{1 + \lambda_{i_0} d_{i_0}}. \tag{31}$$

Waiting time: Consider next the costs associated with the waiting time. The waiting time of the first job is $w_{i_0} = u_{i_0} + 1(u_{i_0} = 0)\, d_{i_0}$, where $d_{i_0} = 0$ if Server i_0 is `NeverOff`. The waiting time of the second job is obtained by subtracting the mean service time $\mathrm{E}[X]/\nu_{i_1}$ from (29),

$$\mathrm{E}[W_{i_1}] = d_{i_1} e^{-\lambda u_{i_1}^*} + u_{i_1}^* - \frac{1 - e^{-\lambda u_{i_1}^*}}{\lambda}.$$

The costs the later arriving jobs incur on average follow from the value function (25), where the backlogs are determined at time T_1. Consequently, in this case the result depends on the first two moments of $U_i(T_1)$. For $i \neq i_1$, we have

$$\mathrm{E}[U_i(T_1)^2] = \int_0^{u_i^*} \lambda e^{-\lambda t}(u_i^* - t)^2\,dt = \frac{2 - 2e^{-\lambda u_i^*} + \lambda u_i^*(\lambda u_i^* - 2)}{\lambda^2}.$$

For i_1, the expression becomes more complicated. Assuming `InstantOff`, we have

$$\begin{aligned}\mathrm{E}[U_{i_1}(T_1)^2 \mid X_{i_1} = x] &= \int_0^{u_{i_1}^*} \lambda e^{-\lambda t}(u_{i_1}^* - t + x)^2\,dt + e^{-\lambda u_{i_1}^*}(x + d_{i_1})^2 \\ &= \frac{2 + (2x\lambda + d_{i_1}(d_{i_1} + 2x)\lambda^2 - 2)e^{-\lambda u_{i_1}^*} + \lambda(u_{i_1}^* + x)(\lambda(u_{i_1}^* + x) - 2)}{\lambda^2}.\end{aligned}$$

Taking the expectation over $X_{i_1} = X/\nu_{i_1}$ then gives

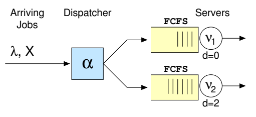
$$\mathrm{E}[U_{i_1}(T_1)^2] = \frac{2 + \lambda u_{i_1}^*(\lambda u_{i_1}^* - 2) + ((\lambda d_{i_1})^2 - 2)e^{-\lambda u_{i_1}^*}}{\lambda^2} + \frac{2\left((1 + \lambda d_{i_1})e^{-\lambda u_{i_1}^*} + \lambda u_{i_1}^* - 1\right)}{\lambda}\frac{\mathrm{E}[X]}{\nu_{i_1}} + \frac{\mathrm{E}[X^2]}{\nu_{i_1}^2}. \tag{32}$$

If the server i_1 is `NeverOff`, then the corresponding first and second moment of $U_{i_1}(T_1)$ are obtained from (29) and (32) by setting $d_{i_1} = 0$. The costs of action (i_0, i_1) with respect to waiting time are thus,

$$L_W(i_0, i_1) \triangleq w_{i_0} + \mathrm{E}[W_{i_1}] + \sum_{j=1}^{m}\frac{\lambda_j}{2(1 - \rho_j)}\left(\mathrm{E}[U_j(T_1)^2] - \frac{d_j(2 + \lambda_j d_j)\,\mathrm{E}[U_j(T_1)]}{1 + \lambda_j d_j}\right). \tag{33}$$

Also here the summation over all queues, $j = 1, \ldots, m$, can be removed, yielding

$$L_W^*(i_0, i_1) \triangleq w_{i_0}^* + \mathrm{E}[W_{i_1}] + \sum_{j \in \cup\{i_k\}}\frac{\lambda_j}{2(1 - \rho_j)}\left(\mathrm{E}[U_j(T_1)^2] - \mathrm{E}[U_j^{(0)}(T_1)^2] - \frac{d_j(2 + \lambda_j d_j)}{1 + \lambda_j d_j}\left(\mathrm{E}[U_j(T_1)] - \mathrm{E}[U_j^{(0)}(T_1)]\right)\right), \tag{34}$$



Server 1:	$\nu_1 = 1$	$e_1 = 1$		`NeverOff`
Server 2:	$\nu_2 = 1$	$e_2 = 1$	$d_2 = 2$	`InstantOff`

Table 2: Two-server system.

where $U_j^{(0)}(T_1)$ denotes the backlog in Queue j at time T_1 given the two jobs were assigned elsewhere. For brevity, we leave the Lookahead costs with respect to the squared waiting time to reader.

Sojourn time: The expected costs with respect to the sojourn time can be determined in a similar fashion. The only difference from the waiting time case is that we need to also include the service time of the first two jobs in the immediate costs. The first job incurs an immediate cost equal to $u^*_{i_0}$, which includes also the possible (remaining) switching delay. The mean sojourn time of the second job is given by (29). Hence, the mean cost of action (i_0, i_1) with respect to the sojourn time is (with $d_{i_1} = 0$ if `NeverOff`)

$$L^*_T(i_0, i_1) \triangleq u^*_{i_0} + \mathrm{E}[U_{i_1}(T_1)] + \sum_{j \in \cup\{i_k\}} \frac{\lambda_j}{2(1-\rho_j)} \left(\mathrm{E}[U_j(T_1)^2] - \mathrm{E}[U_j^{(0)}(T_1)^2] - \frac{d_j(2+\lambda_j d_j)}{1+\lambda_j d_j} \left(\mathrm{E}[U_j(T_1)] - \mathrm{E}[U_j^{(0)}(T_1)] \right) \right), \tag{35}$$

where the summation over all queues, $j = 1, \ldots, m$, has again been removed.

The number of terms in (31), (34) and (35) is constant, whereas in (30) and (33) the number of terms increases linearly as a function of m. The expected total cost is the sum of, e.g., (31) and (35), and Lookahead chooses Queue i_0 with the minimum cost,

$$\alpha_L = \underset{i_0}{\mathrm{argmin}} \left(\min_{i_1} \; L^*_R(i_0, i_1) + L^*_T(i_0, i_1) \right).$$

Finally we note that in the definition of the value function (4), the long-run average costs $r \cdot t$ are subtracted from the expected incurred costs during the same time interval. Here we have not subtracted them during $(0, T_1)$ as T_1 and the corresponding costs, $r\,\mathrm{E}[T_1]$, are the same for each action (i_0, i_1). However, if comparing action (i_0, i_1), e.g., to FPI action i_0, then also the long-run average costs until T_1 should be taken into account (see [54]).

4.5. Simulations

The numerical results given in this section are obtained using a special purpose simulator written in C. For each sample point, we simulated the system with given policies for around one hundred million jobs and then computed the performance measures of interest. Consequently, the analytical results, available, e.g., for the RND policies, match "exactly" with the simulated curves. The basic policy for FPI and Lookahead is SITA-E.

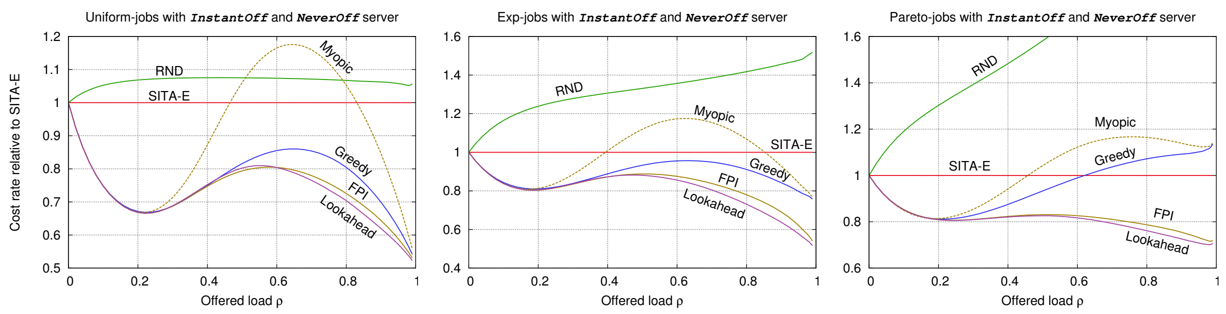
4.5.1. Example 1: Waiting time and running costs

Our first example system comprises two heterogeneous servers with the same service rate ($m = 2$, $\nu_1 = \nu_2 = 1$) but with different switching-off policies: Server 1 is running non-stop (`NeverOff`), whereas Server 2 is switched-off when it becomes idle (`InstantOff`). The switching delay is 2 time units. These and the other system parameters are given in Table 2. In addition to the waiting time, we are also interested in energy consumption so our objective is

$$\min \quad \left(\lambda\,\mathrm{E}[W] + \sum_{i=1}^{m} e_i \cdot \mathrm{P}\{\text{Server } i \text{ is running}\} \right).$$

Note that with respect to waiting time, Greedy and Myopic would be equivalent, but this is no longer the case when the running costs are included. As $\nu_1 = \nu_2$, the service times are equal in both servers, and therefore the minimization of the mean waiting and sojourn times are equivalent. As for job size distributions, we consider three cases:

1. Uniform: $X \sim \mathrm{U}(0.5, 1.5)$; $\mathrm{E}[X] = 1$ and $\mathrm{V}[X] = 1/12$.
2. Exponential: $X \sim \mathrm{Exp}(1)$; $\mathrm{E}[X] = 1$ and $\mathrm{V}[X] = 1$.
3. Pareto: $X \sim \mathrm{BP}(a, b, \alpha)$, where $(a, b) = (0.37, 37)$ define the interval and $\alpha = 1.5$ is the shape parameter; $\mathrm{E}[X] = 1$ and $\mathrm{V}[X] = 27/10$.



(a) Uniform (b) Exponential (c) Truncated Pareto

Figure 5: Mean cost rate with different job size distributions with the joint-objective of mean waiting time and running costs. The FPI policies manage to reduce costs considerably when the job sizes vary more.

The numerical results are depicted in Fig. 5. The x-axis corresponds to the offered load ρ, and the y-axis is the relative mean performance when compared to SITA-E,

$$\frac{\text{Mean cost rate with policy } \alpha}{\text{Mean cost rate with SITA-E}}.$$

We observe that RND is always worse than SITA-E, as expected.[3] Greedy is generally better than Myopic, which suggests that Myopic avoids starting Server 2 unnecessarily. Note also that with Pareto distributed job sizes both are eventually worse than SITA-E (as expected). In contrast, FPI and Lookahead, both based on SITA-E, always achieve the lowest cost rates with Lookahead being slightly better. The difference relative to the other policies is especially striking in Fig. 5(c), where the job sizes vary the most.

4.5.2. *Example 2: Squared waiting time*

Let us consider next the minimization of the mean squared waiting time combined with running costs,

$$\min \quad \left(\lambda\, \mathrm{E}[W^2] + \sum_{i=1}^{m} e_i \cdot \mathrm{P}\{\text{Server } i \text{ is running}\} \right).$$

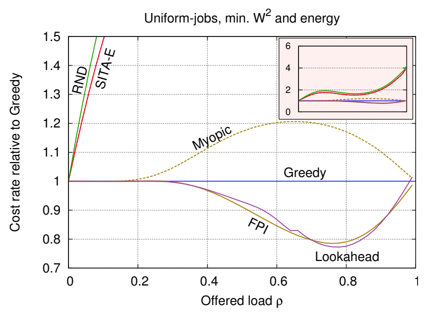
The squared waiting time term combines both efficiency (mean waiting time) and fairness (no job should wait much longer than others). Equivalently, as $\mathrm{E}[W^2] = \mathrm{E}[W]^2 + \mathrm{V}[W]$, the objective is to jointly minimize the mean and the variance of the waiting time, along with the running costs.

We assume the same two-server system, where only the secondary server 2 is switched-off when idle. Note that this is in fact quite a reasonable static compromise because when the load is low, a single server is often sufficient, and at very high level of load the busy periods become long and the importance of the switching delay diminishes automatically. As the performance of SITA-E turns out to be rather weak in this case, we compare the mean performance of each policy to that of Greedy. The numerical results are depicted in Fig. 6, where the small subfigures illustrate the situation at a larger scale. We can observe that the results are quite similar to those of Fig. 5. The dynamic Greedy and Myopic policies are now significantly better than SITA-E and RND in all cases. FPI and Lookahead, based on the value functions derived in Section 3, again yield the best performance in every case. The difference between them is small, except in the case of the Pareto distributed job sizes when ρ is high.

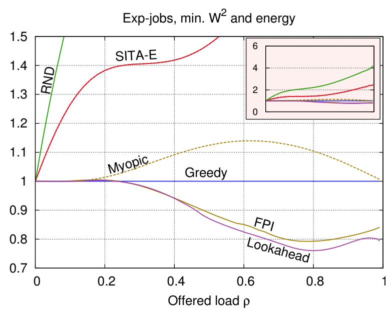
4.5.3. *Example 3: Four servers with `InstantOff` and `NeverOff`*

As a final example, we consider the joint optimization of the server switching-off policy and the dispatching policy. To this end, we assume slightly larger setups with four servers and three different sets of operating parameters, (a)-(c) given in Table 3. In Scenario (a), we have four identical servers with unit service rates $\nu_i = 1$, unit running costs

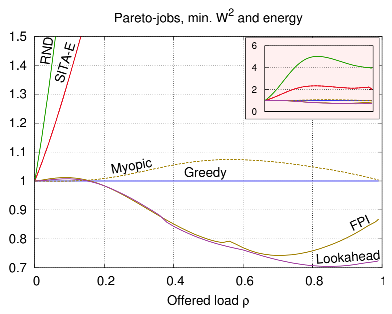
[3]With SITA-E, we assigned the long jobs to Server 2, which leads to longer idle periods, and consequently, to a better performance.



(a) Uniform

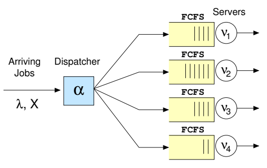


(b) Exponential



(c) Truncated Pareto

Figure 6: Performance in terms of the mean squared waiting time E$[W^2]$ and running costs with different job size distributions. The gap between the static (RND and SITA-E) and the dynamic policies increases.



Parameter		(a) Identical	(b) Linear e	(c) Squared e
service rates	$\nu_1, \ldots, \nu_4$:	$1, 1, 1, 1$	$1, 1, 1, 1$	$1, 2, 3, 4$
running costs	$e_1, \ldots, e_4$:	$1, 1, 1, 1$	$1, 2, 3, 4$	$1, 2, 9, 16$
switching delay	$d_1, \ldots, d_4$:	$1, 1, 1, 1$	$1, 1, 1, 1$	$1, 1, 1, 1$

Table 3: Four-server systems.

$e_i = 1$, and unit switching delays $d_i = 1$. In Scenario (b), the servers are heterogeneous in their energy consumptions, $e_i = i$ (say, older servers are less energy efficient). In Scenario (c), the service rates also vary, $\nu_i = i$, and the energy consumption is assumed to be proportional to the service speed squared, $e_i \propto (\nu_i)^2$ (cf. [40, 55]). The switching delay is the same in every scenario, and the job sizes are exponentially distributed with unit mean.

As we have heterogeneous service rates, it is often more meaningful to consider sojourn time instead of waiting time. Hence, our objective is to minimize the sum of the sojourn times and running costs,

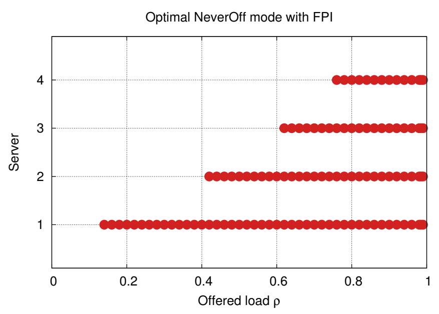
$$\min \quad \left(\lambda \,\mathrm{E}[T] + \sum_{i=1}^{m} e_i \cdot \mathrm{P}\{\text{Server } i \text{ is running}\} \right).$$

Accordingly, the individually optimal Greedy policy considers now the sojourn time, whereas Myopic and FPI are aware of the actual cost structure.

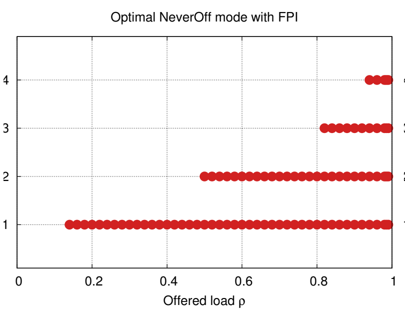
For each dispatching policy and load level ρ, we evaluate all 16 possible configurations of the switching-off policies (`InstantOff` and `NeverOff`) and then choose the best among them. Fig. 7 illustrates the optimal configuration of the switching-off policies as a function of ρ, when the jobs are dispatched using FPI. We can observe that, as expected, initially all servers are in `InstantOff` mode, and as the load increases more and more servers are set to run non-stop (`NeverOff`). Interestingly, in Scenario (c) the slowest server is left in `InstantOff` mode when the load is very high. Note that this does not imply that it is not used, however.

The performance in terms of costs is depicted in Fig. 8. The reference policy is SITA-E with all servers operating under `InstantOff`. We can see that the optimal switching-off strategy can save up to 10% when the dispatching policy remains the same (SITA-E).

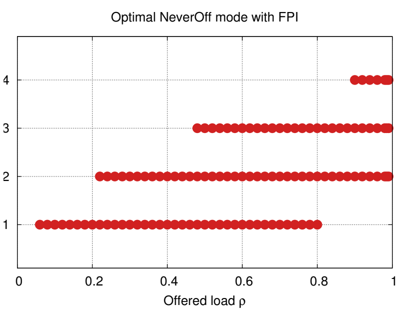
The Myopic policy that minimizes the immediate costs and neglects future arrivals yields a reasonable mean cost rate. A somewhat better performance is obtained by Greedy in Scenarios (a) and (b). However, in the power proportional Scenario (c), Greedy fails miserably, especially when the load is low. This is due to the fact that Greedy favors the fast but expensive server blindly. In contrast, FPI shows a very good and robust performance in each scenario, and shines especially under heavy loads ($\rho > 0.9$) and in Scenario (c), where both Greedy and Myopic have clear problems. Lookahead yields even better performance and is strong even when FPI is marginally behind Greedy in Scenario (a). However, the difference between FPI and Lookahead is not huge, which suggests that FPI is already nearly optimal.



(a) Identical servers

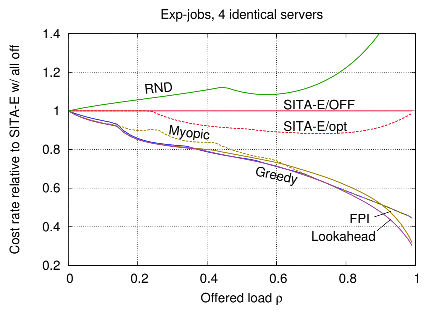


(b) Linear running cost

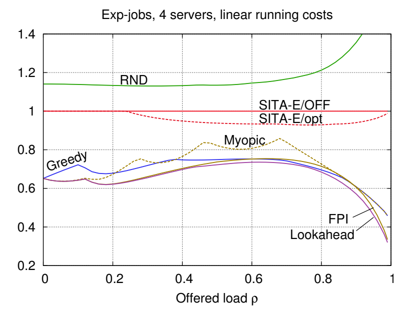


(c) Squared power consumption

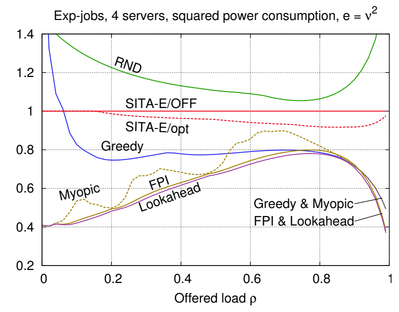
Figure 7: Servers set to `NeverOff` instead of `InstantOff` in the optimal switching-off policy with FPI dispatching policy. Note that the number of always running servers tends to increase as a function of the load.



(a) Identical servers



(b) Linear running cost



(c) Squared power consumption

Figure 8: Numerical results with optimal server-specific switching-off policies for four servers and exponentially distributed jobs.

5. Conclusions

We have studied the classical dispatching or task assignment problem, where arriving jobs are immediately routed to one of several available parallel servers. Because of the current concern about energy consumption, and in contrast to traditional models, we assume that the servers can be switched off in order to save energy. The downside is that there is also a server-specific switching delay, i.e., switching Server i back on takes a certain time d_i (e.g., the boot time of a server). Further, we assume a rather general cost structure, which includes both the server specific running and switching costs, as well as performance-specific metrics such as waiting time and sojourn time, including their second moments (as a measure of fairness).

Our analysis of the value functions for a single M/G/1 queue *with a switching delay* generalizes earlier results that considered only first moments and did not include switching delay. The value functions enable us, by means of FPI and Lookahead, to develop efficient state-dependent dispatching policies, which take into account the switching delay, the cost structure and future arrivals. In the numerical examples, the FPI and Lookahead based policies consistently achieved the lowest mean cost rates. FPI is only the first (albeit often the most significant) step towards the optimal policy. Our future work includes analyzing the settings where the optimal energy-aware dispatching policies can be developed.

Acknowledgements

This work has been partially supported by the Academy of Finland in TOP-Energy project (grant no. 268992).

References

[1] R. Bellman, A Markovian decision process, Indiana Univ. Math. J. 6 (1957) 679–684.
[2] R. A. Howard, Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes, Wiley Interscience, 1971.

[3] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, 2005.
[4] E. Hyytiä, A. Penttinen, S. Aalto, Size- and state-aware dispatching problem with queue-specific job sizes, European Journal of Operational Research 217 (2) (2012) 357–370.
[5] E. Hyytiä, S. Aalto, A. Penttinen, Minimizing slowdown in heterogeneous size-aware dispatching systems, ACM SIGMETRICS Performance Evaluation Review 40 (2012) 29–40, (ACM SIGMETRICS/Performance conference).
[6] A. Penttinen, E. Hyytiä, 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.
[7] A. Ephremides, P. Varaiya, J. Walrand, A simple dynamic routing problem, IEEE Transactions on Automatic Control 25 (4) (1980) 690–693.
[8] Z. Liu, D. Towsley, Optimality of the round-robin routing policy, Journal of Applied Probability 31 (2) (1994) 466–475.
[9] Z. Liu, R. Righter, Optimal load balancing on distributed homogeneous unreliable processors, Operations Research 46 (4) (1998) 563–573.
[10] M. E. Crovella, M. Harchol-Balter, C. D. Murta, Task assignment in a distributed system: Improving performance by unbalancing load, in: Proceedings of SIGMETRICS '98, Madison, Wisconsin, USA, 1998, pp. 268–269.
[11] M. Harchol-Balter, M. E. Crovella, C. D. Murta, On choosing a task assignment policy for a distributed server system, Journal of Parallel and Distributed Computing 59 (1999) 204–228.
[12] H. Feng, V. Misra, D. Rubenstein, Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems, Performance Evaluation 62 (1-4) (2005) 475–492.
[13] W. Winston, Optimality of the shortest line discipline, Journal of Applied Probability 14 (1977) 181–189.
[14] R. R. Weber, On the optimal assignment of customers to parallel servers, Journal of Applied Probability 15 (2) (1978) 406–413.
[15] A. Hordijk, G. Koole, On the optimality of the generalised shortest queue policy, Prob. Eng. Inf. Sci. 4 (1990) 477–487.
[16] D. Towsley, P. Sparaggis, C. Cassandras, Stochastic ordering properties and optimal routing control for a class of finite capacity queueing systems, in: Proc. of the 29th IEEE Conference on Decision and Control, 1990, pp. 658–663.
[17] P. D. Sparaggis, D. Towsley, Optimal routing and scheduling of customers with deadlines, Probability in the Engineering and Informational Sciences 8 (1) (1994) 33–49. doi:10.1017/S0269964800003181.
[18] G. Koole, P. D. Sparaggis, D. Towsley, Minimizing response times and queue lengths in systems of parallel queues, Journal of Applied Probability 36 (4) (1999) 1185–1193.
[19] O. Akgun, R. Righter, R. Wolff, Multiple server system with flexible arrivals, Advances in Applied Probability 43 (2011) 985–1004.
[20] V. Gupta, M. Harchol-Balter, K. Sigman, W. Whitt, Analysis of join-the-shortest-queue routing for web server farms, Performance Evaluation 64 (9-12) (2007) 1062–1081.
[21] W. Whitt, Deciding which queue to join: Some counterexamples, Operations Research 34 (1) (1986) 55–62.
[22] D. J. Daley, Certain optimality properties of the first-come first-served discipline for G/G/s queues, Stochastic Processes and their Applications 25 (1987) 301–308.
[23] S. G. Foss, Approximation of multichannel queueing systems, Siberian Mathematical Journal 21 (1980) 851–857.
[24] D. Stoyan, A critical remark on a system approximation in queueing theory, Math. Operationsforsch. Statist. 7 (1976) 953–956.
[25] K. R. Krishnan, Joining the right queue: a state-dependent decision rule, IEEE Transactions on Automatic Control 35 (1) (1990) 104–108.
[26] S. Aalto, J. Virtamo, Basic packet routing problem, in: The thirteenth Nordic teletraffic seminar NTS-13, Trondheim, Norway, 1996, pp. 85–97.
[27] E. Hyytiä, J. Virtamo, S. Aalto, A. Penttinen, M/M/1-PS queue and size-aware task assignment, Performance Evaluation 68 (11) (2011) 1136–1148, (IFIP Performance'13).
[28] E. Hyytiä, A. Penttinen, S. Aalto, J. Virtamo, Dispatching problem with fixed size jobs and processor sharing discipline, in: 23rd International Teletraffic Congress (ITC'23), San Fransisco, USA, 2011, pp. 190–197.
[29] J. M. Norman, Heuristic procedures in dynamic programming, Manchester University Press, 1972.
[30] S. A. E. Sassen, H. C. Tijms, R. D. Nobel, A heuristic rule for routing customers to parallel servers, Statistica Neerlandica 51 (1) (1997) 107–121.
[31] S. Bhulai, On the value function of the M/Cox(r)/1 queue, Journal of Applied Probability 43 (2) (2006) 363–376.
[32] 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.
[33] J. van Leeuwaarden, S. Aalto, J. Virtamo, Load balancing in cellular networks using first policy iteration, Technical report, Networking Laboratory, Helsinki University of Technology (2001).
[34] J. R. Artalejo, A. Economou, M. J. Lopez-Herrero, Analysis of a multiserver queue with setup times, Queueing Syst. Theory Appl. 51 (1–2) (2005) 53–76.
[35] A. Gandhi, M. Harchol-Balter, M/G/k with exponential setup, Technical Report CMU-CS-09-166, Carnegie Mellon University (2009).
[36] A. Gandhi, M. Harchol-Balter, I. Adan, Server farms with setup costs, Performance Evaluation 67 (11) (2010) 1123–1138.
[37] A. Gandhi, V. Gupta, M. Harchol-Balter, M. Kozuch, Optimality analysis of energy-performance trade-off for server farm management, Performance Evaluation 67 (11) (2010) 1155–1171.
[38] I. Mitrani, Managing performance and power consumption in a server farm, Annals of Operations Research 202 (1) (2013) 121–134.
[39] V. Maccio, D. Down, On optimal policies for energy-aware servers, in: In Proc. of MASCOTS, San Francisco, US, 2013.
[40] A. Wierman, L. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems, in: IEEE INFOCOM, 2009, pp. 2007–2015.
[41] P. D. Welch, On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service, Operations Research 12 (5) (1964) 736–752.
[42] W. Bischof, Analysis of M/G/1-queues with setup times and vacations under six different service disciplines, Queueing Syst. Theory Appl. 39 (4) (2001) 265–301.
[43] M. Hassan, M. Atiquzzaman, A delayed vacation model of an M/G/1 queue with setup time and its application to SVCC-based ATM networks, IEICE Transactions on Communications E80-B (2) (1997) 317–323.
[44] J. Artalejo, A unified cost function for M/G/1 queueing systems with removable server, Trabajos de Investigación Operativa 7 (1992) 95–104.
[45] M. Yadin, P. Naor, Queueing systems with a removable service station, Operational Research Quarterly.

[46] D. P. Heyman, Optimal operating policies for M/G/1 queuing systems, Operations Research 16 (2) (1968) pp. 362–382.
[47] D. Heyman, The T-policy for the M/G/1 queue, Management Science 23 (7) (1977) 775–778.
[48] J. Teghem, Jr., Control of the service process in a queueing system, European Journal of Operational Research 23 (2) (1986) 141–158.
[49] B. Doshi, Queueing systems with vacations – a survey, Queueing Systems 1 (1986) 29–66.
[50] E. Feinberg, O. Kella, Optimality of D-policies for an M/G/1 queue with a removable server, Queueing Systems 42 (4) (2002) 355–376.
[51] R. B. Cooper, Introduction to Queueing Theory, 2nd Edition, North Holland, 1984.
[52] L. Kleinrock, Queueing Systems, Volume I: Theory, Wiley Interscience, 1975.
[53] J. Medhi, Stochastic Models in Queueing Theory, 2nd Edition, Elsevier, 2003.
[54] E. Hyytiä, Lookahead actions in dispatching to parallel queues, Performance Evaluation 70 (10) (2013) 859–872, (IFIP Performance'13).
[55] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proc. of 36th IEEE Annual Symposium on Foundations of Computer Science, 1995, pp. 374–382.
[56] R. Wolff, Poisson arrivals see time averages, Operations Research 30 (1982) 223–231.

Appendix A. PASTA applied

Consider an M/G/1 queue with arrival rate λ, mean service time $\mathrm{E}[S]$, and load $\rho = \lambda\,\mathrm{E}[S] < 1$. Let B_u denote a busy period that starts with an initial backlog of u, and N_u the number of customers that arrive during $(0, B_u)$. The backlog process at time t is denoted by $U(t)$. It is well known that

$$\mathrm{E}[N_u] = \lambda\,\mathrm{E}[B_u] = \frac{\lambda u}{1-\rho}. \tag{A.1}$$

The arrival times and waiting times of these customers are denoted by α_i and W_i, respectively. Thus, we have $W_i = U(\alpha_i^-)$.

Proposition 8. *For any non-negative function $f(x)$ defined on $[0,\infty)$, we have*

$$\mathrm{E}[\sum_{i=1}^{N_u} f(W_i)] = \lambda\,\mathrm{E}[\int_0^{B_u} f(U(t))\,dt].$$

Proof Let $f(x)$ be non-negative function defined on $[0,\infty)$. Construct now a regenerative backlog process $U(t)$ by repeating IID cycles, each distributed as B_u. In other words, as soon as the first busy period with initial backlog u ceases, we start a new one (again with an initial backlog of u) and so on. Since $\mathrm{E}[B_u] < \infty$, we have

$$\bar{U} \triangleq \lim_{t\to\infty} \mathrm{E}[f(U(t))] = \frac{\mathrm{E}[\int_0^{B_u} f(U(t))\,dt]}{\mathrm{E}[B_u]}. \tag{A.2}$$

Let then α_i denote the arrival times of new customers in the construction defined above. In addition, let W_i refer to the corresponding waiting times, $W_i = U(\alpha_i^-)$ for all i. Now (i) the α_i constitute a Poisson process with intensity λ, and (ii) the W_i constitute a regenerative process (in discrete time) with IID cycles distributed as N_u. Since $\mathrm{E}[N_u] < \infty$, we have

$$\bar{W} \triangleq \lim_{i\to\infty} \mathrm{E}[f(W_i)] = \frac{\mathrm{E}[\sum_{i=1}^{N_u} f(W_i)]}{\mathrm{E}[N_u]}. \tag{A.3}$$

On the other hand, due to the PASTA property [56], we have

$$\bar{U} = \bar{W}. \tag{A.4}$$

Thus,

$$\begin{aligned}
\mathrm{E}[\sum_{i=1}^{N_u} f(W_i)] &\overset{(A.3)}{=} \mathrm{E}[N_u]\,\bar{W} \\
&\overset{(A.4)}{=} \mathrm{E}[N_u]\,\bar{U} \\
&\overset{(A.1)}{=} \lambda\,\mathrm{E}[B_u]\,\bar{U} \\
&\overset{(A.2)}{=} \lambda\,\mathrm{E}[\int_0^{B_u} f(U(t))\,dt].
\end{aligned}$$

which completes the proof. ■

Appendix B. General job- and server-specific cost rates

As briefly mentioned at the start of Section 3.3, it is possible to define job-specific holding cost rates. That is, each Job j incurs costs at a possibly increasing rate, denoted by $\omega(t)$, during its waiting and/or sojourn time. Moreover, this holding cost rate can be job-specific, $\omega_j(t)$, so that some high priority jobs can incur costs faster than other jobs. More specifically, we assume that jobs are defined by i.i.d. triples $(X_j, A_j, B_j) \sim (X, A, B)$, where X_j is the size, and

$$\omega_j(t) = A_j + B_j t, \qquad (A_j, B_j \geq 0)$$

is the job-specific holding cost rate. Even though the triples are i.i.d., the size and holding cost parameters of the same job can depend on each other (cf. the slowdown metric where $A_j = 1/X_j$ [5]). Using the results of Section 3.3, we have for the job-specific holding cost during the waiting time in an M/G/1 queue with switching delay d

$$v_J(u) - v_J(0) = \frac{\lambda u}{1-\rho}\left(\frac{\mathrm{E}[B]}{3}u^2 + \left(\frac{\mathrm{E}[A]}{2} + \frac{\lambda\,\mathrm{E}[X^2]}{2(1-\rho)}\right)\left(u - \frac{d(2+\lambda d)}{1+\lambda d}\right) - \frac{d^2(3+\lambda d)}{3(1+\lambda d)}\mathrm{E}[B]\right), \tag{B.1}$$

which is a polynomial of the third degree, $a_3u^3 + a_2u^2 + a_1u + a_0$. In general, a job-specific holding cost rate $\omega_j(t)$ of degree k corresponds to the waiting time W^{k+1} or sojourn time T^{k+1}, which, according to (11), yields a polynomial value function of degree $k+2$ that depends on the $(k+1)$ first moments of the job size X. The job-specific holding cost rates can also depend on the server. Moreover, it is straightforward to define job- and server-specific processing cost rates E_j for the jobs (e.g., when the energy consumption depends on the job and the server). The switching costs may be server specific (as in Section 4), but they are unlikely to depend on the job that starts a busy period.