Control and optimization of the SRPT service policy by frequency scaling

Andrea Marin[2], Isi Mitrani[3], Maryam Elahi[1], and Carey Williamson[1]

[1] University of Calgary, Department of Computer Science, Calgary, Canada
{bmelahi,carey}@ucalgary.ca
[2] Università Ca' Foscari Venezia, DAIS, Venice, Italy, marin@unive.it
[3] Newcastle University, School of Computing, Newcastle, UK
isi.mitrani@newcastle.ac.uk

Abstract. In this paper, we study a system where the speed of a processor depends on the current number of jobs. We propose a queueing model in which jobs consist of a variable number of tasks, and priority is given to the job with the fewest remaining tasks. The number of processor frequency levels determines the dimensionality of the queueing process. The objective is to evaluate the trade-offs between holding cost and energy cost when setting the processor frequency. We obtain exact results for two and three frequency levels, and accurate approximations that can be further generalized. Numerical and simulation results show the high accuracy of the approximate solutions that we propose. Our experiments suggest that a parsimonius model with only two frequency levels is sufficient, since more elaborate models provide negligible improvements when optimizing the system.

1 Introduction

In dynamic speed scaling systems, the speed at which the processor executes jobs is adjusted dynamically based on the workload experienced by the system. Modern processors typically support over a dozen discrete operating speeds, often with a factor of two (or more) between the slowest and the fastest speeds available.

Multiple tradeoffs exist in such speed scaling systems. The most obvious is the tradeoff between response time and energy consumption (see. e.g., [17, 4, 13]). To minimize response time, one would use the highest system speed available, while to minimize energy consumption, one would use the lowest system speed. For this reason, most speed scaling research uses a cost function that combines response time (i.e., job delay, or holding cost) and energy consumption when doing system optimization as in [16]. Another interesting tradeoff arises from the interaction between the job scheduling policy and the speed scaling function. In job-count-based speed scaling, for instance, the CPU speed is set dynamically based on the current number of jobs in the system. As a result, different scheduling policies produce different costs, since the average number of jobs in the system varies. For example, Shortest Remaining Processing Time (SRPT) minimizes system occupancy, and thus tends to run at lower speeds and for longer times than other scheduling policies, such as Processor Sharing (PS) [1].

Another consequence of SRPT-based scheduling is extreme unfairness to large jobs. There are two underlying reasons for this unfairness. First, large jobs tend to wait much longer to receive service, because of SRPT's bias toward short jobs. Second, even when they do receive service, large jobs tend to be served at low(er) speeds, since there are usually very few jobs (perhaps only one) in the system at that point [14].

Size based scheduling disciplines have been considered of primary importance in queueing theory (see, e.g., [9, Ch. 3]) and for practical applications in computer networking (see, e.g., [11] and the references therein). Despite the problems concerning the fairness [3, 1], the optimality of SRPT makes it practically appealing for the situations where job sizes can be predicted accurately. This is the case, for example, of TCP flows whose size is known in advance, e.g., in transferring static resources from a web server as shown in [8].

In this paper, we further investigate performance tradeoffs in dynamic speed scaling systems. One basic dilemma in these systems is what speed to use when there is only a single job in the system. Should it run at the highest speed, to minimize delay, or at the lowest speed, to conserve energy? To answer this question, we investigate a model in which we can determine the optimal speeds to use within a finite set of available speeds.

The main contributions in this paper are the following:

1. we propose a novel analytical model for dynamic speed scaling systems that use the SRPT scheduling policy, with K available speeds;
2. we derive exact analytic results for $K = 2$ to optimize the system speeds and minimize the system cost function. The approach can be extended to deal with the case $K = 3$;
3. we derive approximate analytic results for $K = 2$ that can be generalized for larger K;
4. we conduct numerical and simulation experiments to verify the accuracy of our analytical models and we provide new insights on the importance of the available speeds in dynamic speed scaling systems with SRPT scheduling.

The rest of this paper is organized as follows. Section 2 provides a brief summary of prior related work on dynamic speed scaling systems. Section 3 presents our system model. Section 4 derives our exact and approximate models for $K = 2$ which may be extdened to more general cases. Section 5 presents numerical results to evaluate the models and to quantify the benefits of frequency scaling. Finally, Section 6 concludes the paper.

2 Related work

SRPT is a preemptive policy that always selects for service the pending job in the system with the least remaining work. In single-speed systems, SRPT is optimal for mean response time [12]. Although SRPT minimizes the mean response time, it is rarely used in practice, since it can be unfair. In particular, large jobs may starve if small jobs have precedence.

Prior literature on speed scaling systems appears in both the theory and systems communities. The theoretical work typically focuses on formal mathematical proofs of the properties of speed scaling systems, such as optimality and fairness. Systems research typically focuses on robust solutions, rather than optimal ones. In this literature review, we focus primarily on the theoretical work, as relevant background context for our paper.

In speed scaling systems, there are many tradeoffs between service rate, response time, energy consumption, and fairness. Yao et al. [17] conducted one of the first analytical studies of dynamic speed scaling systems in which jobs have explicit deadlines, and the service rate is unbounded. Bansal et al. [4] considered an alternative approach that minimizes system response time, within a fixed energy budget. Other work has focused on finding the optimal fixed rate at which to serve jobs in a system with dynamically-settable speeds [7, 15, 16].

Energy-proportional speed scaling is a prevalent approach, which is nearly optimal [1, 2]. In this model, the power consumption $P(s)$ of the system depends on the speed s, which in turn depends on the number of jobs in the system. Bansal, Chan, and Pruhs [2] showed that SRPT with the speed scaling function $P^{-1}(n+1)$ is 3-competitive for an arbitrary power function P. Andrew et al. [1] showed that the optimal policy is SRPT with a job-count-based speed scaling function of the form $s = P^{-1}(n\beta)$.

Fairness in dynamic speed scaling systems is also an important consideration. In particular, speed scaling systems face inherent tradeoffs between fairness, robustness, and optimality [1]. Processor Sharing (PS) is always fair, even under speed scaling. However, the unfairness of SRPT is magnified under speed scaling, since large jobs tend to run at lower speeds (i.e., when the system is mostly empty). Although PS is fair, it is suboptimal for response time and energy [1].

In this paper, we focus on SRPT scheduling with job-count-based speed scaling. Our model builds upon ideas from Andrew *et al.* [1], as well as recent work by Elahi et al. on the autoscaling properties of dynamic speed scaling systems [6]. We derive exact and approximate models to facilitate optimization of the system cost, by determining the optimal service rates to use.

3 Description of the model

In this section we introduce the queueing model that we consider in this paper together with the associated notation (a summary is given in Table 1). Jobs arrive into the system in a Poisson stream with rate λ, and are served by a single server. Each job consists of a random non-empty batch of i.i.d. service phases which will be referred to as *tasks*. The duration of each task, if served at speed 1 instruction per second, is distributed exponentially with mean 1. The number of tasks in a job's batch will be referred to as the *size* of the job. Those sizes are i.i.d. random variables with an arbitrary distribution: a job has size

λ	Intensity of the arrival process
q_i	Prob. distribution of the number of tasks in a job
Q	Finite average job size
K	Number of frequency levels
μ_k	Service rate at frequency level k
u_k	Stationary probability that frequency level k is operating
L	Expected number of tasks in the system
$\mathbf{n} = (n_1, \ldots, n_K)$	State of the system
$p_k(n)$	Stationary probability of observing n tasks at level (queue) k
$w_k(z)$	Generating function of $p_k(n)$
$a(z)$	Generating function of q_i
r_i	Probability of a job size strictly larger than i
$b(z)$	Generating function of r_i
$\pi_k(n)$	Marginal stationary probability of observing n tasks in queue k
$\pi(n_1, \ldots, n_K)$	Stationary probability of state $(n_1, \ldots, n_K)$

Table 1: Summary of the notation used in the paper.

i with probability q_i ($i = 1, 2, \ldots$). The average job size, Q, is assumed to be finite.

This job composition means that the possible distributions of job *lengths* (i.e. the sums of their constituent tasks), belong to a large sub-class of the Coxian distributions (see [5]), which are known to be quite general for practical purposes.

The job scheduling policy is a version of SRPT based on remaining sizes, rather than lengths. That is, at any moment, the job with the smallest number of remaining tasks is served. That policy is combined with a control mechanism whereby the frequency of the processor, i.e. the speed at which it works, is scaled according to the current load. There are K possible frequency levels. If there is only 1 job present, it is served at rate μ_1 tasks per unit time; if there are 2 jobs, then the one with fewer remaining tasks is served at rate μ_2 tasks per unit time, with $\mu_2 > \mu_1$; $\ldots$; if there are K or more jobs present, then the one with the fewest remaining tasks is served at rate μ_K tasks per unit time, with $\mu_K > \mu_{K-1}$.

One is faced with the question of how best to choose the frequency levels. Clearly there are trade-offs between the costs of increasing the processor speed and the costs of holding tasks in the system. We address that question by introducing a cost function which has two components: a cost proportional to the average number of tasks remaining across jobs present, L, and a cost proportional to the average square (see, e.g., [16]) of the service rate:

$$C = c_1 L + c_2 \sum_{k=1}^{K} u_k \mu_k^2 \, , \tag{1}$$

where c_1 and c_2 are given coefficients, and u_k is the probability that frequency level k is in operation. We assume that when the system is empty, its power consumption is that corresponding to the lowest operating speed μ_1.

The purpose of the subsequent analysis is to provide algorithms for computing L and u_k, and hence evaluate the cost function for a given set of parameters. The optimal values of μ_k can then be found by applying an appropriate numerical search.

The operation of the scheduling policy can be modeled by using K task queues, numbered 1, 2, …, K. Sort the jobs present in the system in decreasing order of the numbers of their remaining tasks. Then queue 1 contains the remaining tasks of job 1 (the largest in the system), queue 2 contains the remaining tasks of job 2, if any, …, queue $K-1$ contains the remaining tasks of job $K-1$, if any, and queue K contains the remaining tasks of all other jobs, if any, whose sizes are smaller than that in queue $K-1$. Queue K is the only one where there may be tasks from more than one job. Moreover, the number of tasks in queue i is always larger or equal to those in queue j if $i < j$ and $1 \leq i, j \leq K-1$, while queue K can contain an arbitrary number of tasks. At any epoch, only the tasks from the non-empty queue with the largest index are served with the speed associated with that queue. Therefore, the operating frequency of the processor depends on the number of jobs in the system and corresponds to speed μ_i when there are i jobs in the system, for $i = 1, \dots, K-1$ and is μ_K if there are K or more jobs, as required.

The state of the system at a given moment in time is a vector $(n_1,n_2,\dots,n_K)$, specifying the contents of the K queues. That vector satisfies $(n_1 \geq n_2 \geq \cdots \geq n_{K-1})$, but it is possible that $n_K > n_{K-1}$. The server always serves the non-empty queue with the largest index, and if that index is i, it works at the rate of μ_i tasks per unit time.

An incoming job of size s tasks which finds the system in state $(n_1,n_2,\dots,n_K)$ may cause a reassignment of tasks to queues in order to preserve the shortest-remaining order. If $s \leq n_{K-1}$, then no such reassignment is necessary and the resulting state is $(n_1, \dots, n_{K-1}, n_K + s)$. Otherwise, the incoming s tasks replace the content of queue i, where i is the lowest index such that $s > n_i$. Queue K receives the tasks from queue $(K-1)$, so that the new state is $(n_1, \dots, n_{i-1}, s, n_i, \dots, n_K + n_{K-1})$. If $s > n_1$, then the part of the vector preceding s is empty.

Consider now the steady-state probability, $p_k(n)$, that the total number of tasks in queues 1, 2, …, k is n, while queues $k+1, \dots, K$ are empty ($k = 1, 2, \dots, K$). For every k, $p_k(0)$ is the probability of an empty system, so we may sometimes omit the index and just write $p(0)$. The difference $p_k(n) - p_{k-1}(n)$, for $k = 1, 2, \dots, K$ and $n > 0$, with $p_0(n) = 0$ by definition, is the probability that there are n tasks present and the non-empty queue with the highest index is queue k. In other words, that is the probability that there are n tasks present and the service rate is μ_k. These probabilities are 0 when $n < k$.

Let $w_k(z)$ be the generating function of $p_k(n)$:

$$w_k(z) = \sum_{n=0}^{\infty} z^n p_k(n) \;\; ; \;\; k = 1, \dots, K \; .$$

The last of these functions, $w_K(z)$, corresponds to the distribution of the total number of tasks in the system. It satisfies the normalizing condition $w_K(1) = 1$.

The marginal probabilities, u_k, that the server works at rate μ_k (they appear in the cost function (1)), are given by

$$u_1 = w_1(1) \;\; ; \;\; u_k = w_k(1) - w_{k-1}(1) \;\; ; \;\; k = 2, \ldots, K \; .$$

Let $a(z)$ be the generating function of the job size distribution:

$$a(z) = \sum_{n=1}^{\infty} z^n q_n \; .$$

We shall also need the excess probabilities, r_n, that the size of a job is strictly greater than n, for $n = 0, 1, \ldots$ ($r_0 = 1$). The generating function of r_n, $b(z)$, is given by

$$b(z) = \sum_{n=0}^{\infty} z^n r_n = 1 + \sum_{n=1}^{\infty} z^n [1 - \sum_{j=1}^{n} q_j] \; .$$

The following relation exists between $b(z)$ and $a(z)$:

$$b(z) = \frac{1 - a(z)}{1 - z} \; . \tag{2}$$

This is established by expanding $b(z) - zb(z)$ and performing cancellations. Note that the value of $b(z)$ at $z = 1$ is the average job size: $b(1) = a'(1) = Q$.

We have the following result.

Lemma 1. *The generating functions $w_1(z)$, $w_2(z)$, …, $w_K(z)$ satisfy the relation*

$$w_K(z)[\mu_K - \lambda z b(z))] = \mu_1 p(0) + \sum_{k=1}^{K-1} (\mu_{k+1} - \mu_k) w_k(z) \; . \tag{3}$$

Setting $z = 1$ in (3) yields the normalization condition:

$$w_K(1) = \frac{\mu_1 p(0) + \sum_{i=1}^{K-1} (\mu_{i+1} - \mu_i) w_i(1)}{\mu_K - \lambda Q} = 1 \; . \tag{4}$$

The following proposition gives the necessary and sufficient conditions for the stability of the system.

Proposition 1. *The queueing system is stable if and only if*

$$\lambda Q < \mu_K \; . \tag{5}$$

The proof uses the Pakes' lemma [10] which gives sufficient conditions for stability of Markov chains.

The steady-state average number of tasks in the system, L, is given by $w'_K(1)$. First, by differentiating (2) and applying De L'Hôpital's rule at $z = 1$, we find that $b'(1) = -a''(1)/2$. Now, taking the derivative of (3) at $z = 1$ and rearranging terms, we obtain

$$L = \frac{\sum_{i=1}^{K-1}(\mu_{i+1} - \mu_i)w'_i(1) + \lambda(Q + \frac{1}{2}a''(1))}{\mu_K - \lambda Q} . \tag{6}$$

Thus the quantities that are needed in order to evaluate the cost function (1) are expressed in terms of the functions $w_i(z)$ for $i = 1, 2, \dots, K-1$, i.e. in terms of the probabilities of all states in which queue K is empty. In the following sections we provide exact and approximate solutions for those probabilities, in the cases $K = 2$ and $K = 3$. The methodology employed can be extended to deal with higher values of K, at the price of increased complexity.

The following general result will be useful. Denote by $\pi_1(i)$ the marginal probability that there are i tasks in queue 1 (and any numbers in the other queues). Then

$$\lambda r_n \sum_{i=0}^{n} \pi_1(i) = \mu_1 \pi(n+1, 0, \dots, 0) , \tag{7}$$

where $\pi(n+1, 0, \dots, 0)$ is the probability that queue 1 contains $n+1$ tasks and all other queues are empty. This equation is obtained by balancing the flows across a cut that separates the set of states with no more than n tasks in queue 1, from all other states.

The probabilities $\pi_1(i)$ can also provide an expression for the expected residence time, T_1, of a job in queue 1. Indeed, if the system is in state $(i, \cdot)$, then jobs arrive into queue 1 at rate λr_i. On the other hand, the average number of jobs in queue 1 (not tasks!) is equal to the probability that queue 1 is not empty, i.e., $1 - p(0)$. Therefore, according to Little's result:

$$T_1 = \frac{1 - p(0)}{\lambda \sum_{i=0}^{\infty} r_i \pi_1(i)} . \tag{8}$$

4 The model with $K = 2$ frequency levels

In this section we consider our model with two frequency levels, i.e., the server works at speed μ_1 when there is only one job in the system, and μ_2 when there are at least two jobs, with $\mu_1 < \mu_2$. In Section 4.1 we provide the exact solution of the model, whereas in Section 4.2 we give an approximate, yet a more efficient approach to the computation of the stationary performance indices. The accuracy of the approximation will be assessed in Section 5 and will be shown to be very high.

4.1 Exact analysis

The detailed system state is now a pair (i, j), where i is the number of tasks in queue 1 and j is the number of tasks in queue 2. A service completion causes a transition from state $(i, 0)$ to state $(i-1, 0)$ at rate μ_1 $(i > 0)$, and from state (i, j) to state $(i, j-1)$ at rate μ_2 $(j > 0)$. An arrival of a job of size k causes a transition from state (i, j) to state $(i, j+k)$ if $k \le i$, and to state $(k, j+i)$ if $k > i$. The states $(0, j)$, for $j > 0$, are unreachable and their probabilities are 0.

Denote the probability of state (i, j) by $\pi(i, j)$. These probabilities satisfy the following global balance equations:

$$\lambda\pi(0,0) = \mu_1\pi(1,0) \ . \tag{9}$$

$$(\lambda+\mu_1)\pi(i,0) = \lambda q_i\pi(0,0) + \mu_1\pi(i+1,0) + \mu_2\pi(i,1) \ . \tag{10}$$

$$(\lambda+\mu_2)\pi(i,j) = \lambda\sum_{k=1}^{m} q_k\pi(i,j-k) + \lambda q_i\sum_{k=1}^{m_1}\pi(k,j-k) + \mu_2\pi(i,j+1) \ , \tag{11}$$

where $m = \min(i, j)$ and $m_1 = \min(i-1, j)$.

Since all solutions of these equations are proportional to each other, it is enough to find one solution. The values $\pi(i, j)$ can then be normalized by dividing each of them by their sum.

Start by setting $\pi(0, 0) = 1$. Equation (9) then gives $\pi(1, 0) = \rho_1$, where $\rho_1 = \lambda/\mu_1$. Note that this quantity does not represent offered load, since λ is the arrival rate of jobs, while μ_1 is a service rate of tasks.

Consider the probabilities $\pi(1, j)$, for $j = 0, 1, \ldots$, and define the generating function

$$g_1(z) = \sum_{j=0}^{\infty}\pi(1,j)z^j \ . \tag{12}$$

When $i = 1$ and $j = 0$, equation (10) can be rewritten as

$$(\lambda+\mu_2)\pi(1,0) = \lambda q_1\pi(0,0) + (\mu_2-\mu_1)\pi(1,0) + \mu_1\pi(2,0) + \mu_2\pi(1,1) \ .$$

For $i = 1$ and $j \ge 1$, equations (11) are

$$(\lambda+\mu_2)\pi(1,j) = \lambda q_1\pi(1,j-1) + \mu_2\pi(1,j+1) \ .$$

Multiplying the above by z^j and summing, we get after a little manipulation (and remembering that $\pi(0, 0) = 1$),

$$d_1(z)g_1(z) = \lambda q_1 z - [\mu_1 z + \mu_2(1-z)]\pi(1,0) + \mu_1 z\pi(2,0) \ , \tag{13}$$

where

$$d_1(z) = \lambda z(1-q_1 z) + \mu_2(z-1) \ .$$

This expression for $g_1(z)$ contains an unknown constant, $\pi(2, 0)$. However, note that the coefficient $d_1(z)$ is negative for $z = 0$ and positive for $z = 1$. Therefore, there is a value, z_1, such that $d_1(z_1) = 0$. Since $g_1(z)$ is finite on the whole interval $z \in [0,1]$, the right-hand side of (13) must vanish at $z = z_1$. This gives

$$\mu_1 z_1 \pi(2,0) = [\mu_1 z_1 + \mu_2(1 - z_1)]\pi(1,0) - \lambda q_1 z_1 \pi(0,0) \ .$$

The next step is to consider the probabilities $\pi_{2,j}$, for $j \geq 0$, and their corresponding generating function

$$g_2(z) = \sum_{j=0}^{\infty} \pi(2,j) z^j \ .$$

Repeating the manipulations that led to (13), we obtain

$$d_2(z)g_2(z) = \lambda q_2 z - [\mu_1 z + \mu_2(1-z)]\pi(2,0) + \mu_1 z \pi(3,0) + \lambda q_2 z^2 g_1(z) \ , \quad (14)$$

where

$$d_2(z) = \lambda z(1 - q_1 z - q_2 z^2) + \mu_2(z - 1) \ .$$

Again, the coefficient of $g_2(z)$ is negative at $z = 0$ and positive at $z = 1$. Therefore, there is a value z_2 in the interval (0,1), such that $d_2(z_2) = 0$. Equating the right-hand side of (14) to 0 at $z = z_2$, determines the single unknown constant in that equation, $\pi(3,0)$:

$$\mu_1 z_2 \pi(3,0) = [\mu_1 z_2 + \mu_2(1-z_2)]\pi(2,0) - \lambda q_2 z_2 \pi(0,0) - \lambda q_2 z_2^2 g_1(z_2) \ .$$

The i'th step in this process evaluates the generating function of the probabilities $\pi(i,j)$, $g_i(z)$, in terms of the already known functions $g_1(z)$, $g_2(z)$, …, $g_{i-1}(z)$, and constants $\pi(1,0)$, $\pi(2,0)$, …, $\pi(i,0)$:

$$d_i(z)g_i(z) = \lambda q_i z - [\mu_1 z + \mu_2(1-z)]\pi(i,0) + \mu_1 z \pi(i+1,0) + \lambda q_i z \sum_{k=1}^{i-1} z^k g_k(z) \ , \quad (15)$$

where

$$d_i(z) = \lambda z(1 - \sum_{k=1}^{i} q_k z^k) + \mu_2(z-1)$$

The coefficient of $d_i(z)$ has a zero, z_i, in the interval (0,1), which determines the new unknown constant $\pi(i+1,0)$.

These iterations continue until $g_i(1) < \epsilon$, for some sufficiently small ϵ, or until the largest possible value of i, if the job sizes are bounded. Eventual termination is guaranteed if the queuing process is stable. At that point, all (unnormalized) probabilities $\pi(i,0)$, and hence the function $w_1(z)$, have been determined.

The normalization constant, G, is given by (4):

$$G = \frac{\mu_1 \pi(0,0) + (\mu_2 - \mu_1) w_1(1)}{\mu_2 - \lambda Q} \ . \quad (16)$$

Dividing all $\pi(i,j)$ values, and hence $w_1(1)$, by G, completes the computation of the joint probability distribution of the states (i,j). Lemma 1 now provides $w_2(z)$.

The total average number of tasks in the system, L, is given by (6), which now has the form

$$L = \frac{(\mu_2 - \mu_1)w_1'(1) + \lambda[Q + \frac{1}{2}a''(1)]}{\mu_2 - \lambda Q} .$$

The probabilities that the processor speed is μ_1, and μ_2, are $w_1(1)$ and $1-w_1(1)$, respectively.

4.2 Approximate solution

In order to derive the approximation, consider the marginal probabilities, $\pi(i,\cdot) = g_i(1)$, that there are i tasks in queue 1, with $\pi(0,\cdot) = \pi(0,0)$. These probabilities appear in the left-hand side of (7).

The fraction of time that the system spends in state $(i,\cdot)$, such that queue 2 is not empty, consists of the services of all tasks that join queue 2 when queue 1 reaches size i. This can happen when (a) the system is in state $(k,\cdot)$, for $1 \le k < i$, and a job of size i arrives (in which case k tasks are transferred to queue 2), or (b) the system is in state $(i,\cdot)$ and a job of size k arrives, for $k \le i$; all of its tasks join queue 2. Hence we can write

$$\pi(i,\cdot) - \pi(i,0) = \lambda \left[q_i \sum_{k=1}^{i-1} k\pi(k,\cdot) + \pi(i,\cdot) \sum_{k=1}^{i} kq_k \right] \frac{1}{\mu_2} \; ; \; i = 1, 2, \ldots \; . \tag{17}$$

The first sum in the right-hand side is absent when $i = 1$.

Introducing the notation

$$s_i = \sum_{k=1}^{i} k\pi(k,\cdot) \; ; \; a_i = \sum_{k=1}^{i} kq_k \; , \tag{18}$$

we can rewrite (17) as

$$\pi(i,\cdot) = \frac{\pi(i,0) + q_i \rho_2 s_{i-1}}{1 - \rho_2 a_i} \; ; \; i = 1, 2, \ldots \; , \tag{19}$$

where $\rho_2 = \lambda/\mu_2$ and $s_0 = 0$ by definition.

Start with $\pi(0,0) = 1$ and $\pi(1,0) = \rho_1$. Compute $\pi(1,\cdot)$ from (19), then $\pi(2,0)$ from (7), $\pi(2,\cdot)$ from (19), $\pi(3,0)$ from (7) and so on, up to the desired accuracy. Normalize, using (16).

This procedure is more economical and more stable than the exact solution.

It is worth of notice that both the exact and the approximate approach can be extended to handle $=3$ levels of frequency. However, the complexity of the analysis and its computational cost increase. The approximate approach, instead, can easily be further generalized to models with more than three queues.

The accuracy of the approximation will be examined in Section 5

5 Numerical and simulation results

In this section we describe several experiments aimed at evaluating the accuracy of the approximate solutions that have been proposed, and also observing the behaviour of the cost function (1). Systems with two and three frequency levels are examined. A remarkable observation emerging from these experiments is that, for the purposes of optimization, the models with $K > 2$ may be neglected.

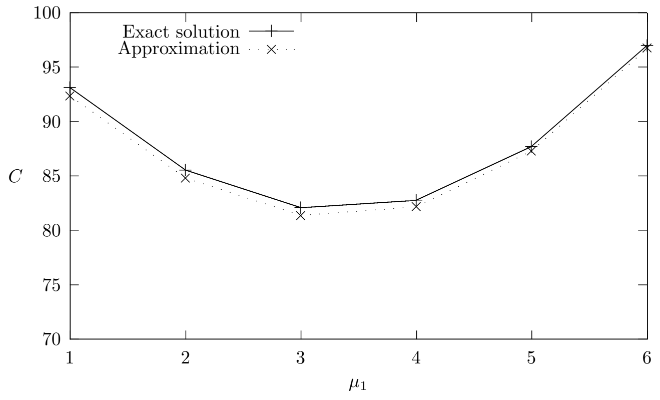

Fig. 1: $K = 2$: cost against μ_1

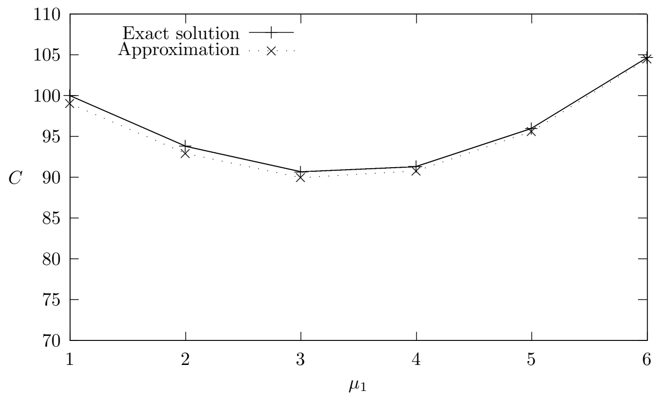

Fig. 2: $K = 2$: Large variance of job sizes

$K = 2$ frequency levels. We consider the model studied in Section 4. In Figure 1, the cost function C is computed exactly and approximately, as de-

Fig. 3: $K = 2$: Impact of the workload intensity on the optimal frequency

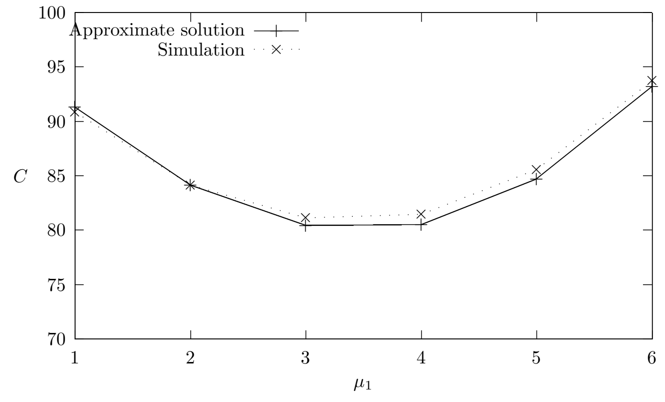

Fig. 4: $K = 3$: cost against μ_1

scribed in section 4, and is plotted against the queue 1 service rate, μ_1. The two cost coefficients are $c_1 = 1$ and $c_2 = 2$. The job arrival rate and the queue 2 service rate are fixed at $\lambda = 1$ and $\mu_2 = 7$. A geometric distribution of job sizes is assumed, with mean 5, truncated at a maximum job size of 50. Thus the offered load, $\lambda Q/\mu_2$ represents about 71% utilization. The value of μ_1 is varied between 1 and 6, in increments of 1. The cost function is convex in μ_1. We have no formal proof of this, but it is invariably observed to be the case. Intuitively, at low values of μ_1 the holding costs dominate, while at large values the service rate costs dominate. Moreover, if a point is reached such that an increase in μ_1 leads to an increase in C, then clearly any further increase in μ_1 would make matters worse. The two plots are very close. The approximate solution underestimates C slightly, but the relative errors never exceed 1%. In particular, both solutions agree that the optimal value of μ_1 is 3 (subject to the granularity of the increments). We next examine the effect of increasing the variance of the job size distribution. Consider a rather extreme case where jobs have size 1, with probability 3/4, or size 17, with probability 1/4. The average job size is the same as in figure 1, $Q = 5$, but the variance has gone up from 20 to 337. All other parameters are the same, and again the cost function is plotted against the service rate μ_1. Figure 2 shows that the increase in variance has led to an increase in costs of between 7% and 10%. The approximate solution is still within less than 1% of the exact one. The optimal value of μ_1 has not changed. The effect of the offered load on the shape of the cost function is illustrated in Figure 3. Three loading regimes are considered, light, moderate and heavy. These are represented by the arrival rates $\lambda = 0.3$, $\lambda = 0.8$ and $\lambda = 1.3$; they correspond to utilizations of about 21%, 57% and 93%, respectively. The other parameters are the same as in Figure 1. Costs are evaluated exactly and are plotted against the queue 1 service rate, μ_1. The figure suggests the following observations, all of which are quite intuitive. As the offered load increases, (a) costs increase; (b) the relative difference between the optimal and the pessimal costs decreases; (c) the optimal value of μ_1 increases.

$K = 3$ frequency levels. The next example evaluates the accuracy of the approximation for a system with three frequency levels. This time μ_2 and μ_3 are fixed at 6 and 7 tasks per second respectively, while μ_1 is varied between 1 and 6, at increments of 1. The job size distribution is geometric with mean 5, and the other parameters are the same as before. Rather than implementing the exact solution, Figure 4 compares the approximated costs with simulated ones. Each simulated point represents a run where $500,000$ jobs arrive into the system (i.e., an average of 2.5 million tasks are served). The approximation is again very accurate, with relative errors not exceeding 1%. Moreover, the two plots agree on the optimal point of $\mu_1 = 3$ (subject to the granularity of the increments). However, the difference in costs between $\mu_1 = 3$ $\mu_1 = 4$ is very slight.

The benefits of optimization. The last experiment attempts to quantify the benefits of optimization, in the context of a 3-queue system under different loading conditions. Three policies are compared. The 'default' policy, or policy 0, does not optimize; it serves all three queues at rate μ_3. Under policy 0, the system is equivalent to a single queue with batch arrivals. Policy 1 serves queues 2 and 3 at rate μ_3, but uses the optimal value for μ_1 (found by a one-dimensional search). This amounts to an optimized $K = 2$ system. Policy 2 serves queue 3 at rate μ_3, but uses the optimal values for μ_1 and μ_2 (found by a two-dimensional search).

For consistency, all costs in this experiment were evaluated by applying the 3-queue approximation. We have relied on the established accuracy of that approximation. A feasible alternative would have been to evaluate policy 0 and policy 1 exactly (using the exact solutions for the cases $K = 1$ and $K = 2$), and resort to approximation only for policy 2.

In Figure 5, the costs incurred by the above three policies are plotted against the job arrival rate λ. It varies between 0.2 and 1.2, while the top service rate remains fixed at $\mu_3 = 7$. Job sizes are distributed geometrically with mean 5, which means that the system loading varies between 14% and 86%. The cost coefficients are $c_1 = 1$ and $c_2 = 2$. When searching for the best μ_1 and μ_2 values, the latter were incremented in steps of 0.5.

The results displayed in Figure 5 are quite instructive. First, we observe that there is less to be gained by optimizing a heavily loaded system, than a lightly loaded one. Of course this was to be expected, since the holding costs become dominant under heavy loads, and minimizing those costs requires large service rates.

The second observation is not so predictable: it seems that the big gains are obtained by optimizing just with respect to μ_1 (policy 1). The additional improvement achieved by optimizing with respect to μ_2 as well (policy 2), is quite minor. It is even debatable whether the expense of searching for policy 2 is justified by the benefits that it brings.

This last observation has practical importance. It suggests that the 2-queue model, rather than being just the simplest special case, is in fact a really significant model from the point of view of control and optimization. One may restrict the search for an optimal policy to the case $K = 2$, and be reasonably confident that the resulting policy would not be bettered by much.

To check whether the above conclusion remains valid under different cost structures, we have repeated the last experiment with several pairs of coefficients c_1 and c_2. Figure 6 shows one such example, where $c_1 = 2$ and $c_2 = 1$ (i.e., holding tasks in the system incurs higher penalties than speeding up the server). The other parameters remain unchanged, and the costs of policies 0, 1 and 2 are plotted against the arrival rate, λ. The conclusion that the most important model is $K = 2$, is confirmed.

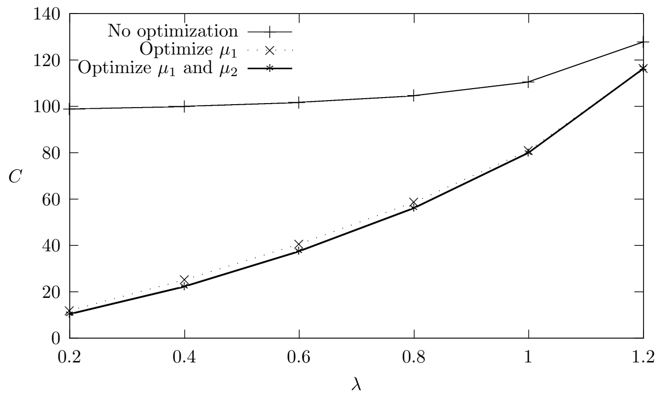

Fig. 5: The benefits of optimization for $c_1 = 1$ and $c_2 = 2$

6 Conclusion

In this paper we studied systems employing the Shortest Remaining Processing Time scheduling discipline with frequency scaling. We have devised a model in which jobs consist of tasks whose service time are exponentially distributed. The distribution of the number of tasks in a job is arbitrary and this allows for a great flexibility in modelling the jobs' service time distribution and the accuracy of its estimation done by the system. The operating frequency is decided on the basis of the number of jobs in the system. The model characteristics allow us to study some important performance indices such as the expected number of tasks in the system, the probability of observing the system operating at a certain frequency level, the expected time that a job remains the largest job in the system and its expected size. We have introduced a cost function that takes

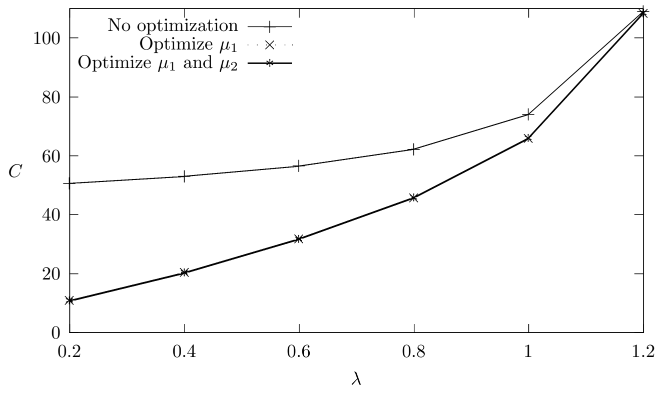

Fig. 6: The benefits of optimization for $c_1 = 2$ and $c_2 = 1$

into account the system's power consumption and the expected number of tasks in the system (and hence the expected response time). We focused the attention on the systems with 2 and 3 frequency levels and showed that the gain in the cost function of the latter with respect to the former is very small, whereas it is high between the system with 2 frequency levels and the one that does not apply any frequency scaling. This suggests that a simple system with 2 frequency levels, opportunely parametrised, can approximately give the benefits in terms of energy saving and quality of service of more complex systems.

Acknowledgments: The work described in this paper has been partially supported by the Università Ca' Foscari Venezia - DAIS within the IRIDE program.

References

1. L. Andrew, M. Lin, and A. Wierman. Optimality, fairness, and robustness in speed scaling designs. In *Proc. of ACM SIGMETRICS*, pages 37–48, 2010.
2. N. Bansal, H. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, 2009.
3. N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling: Investigating unfairness. In *Proc. of ACM SIGMETRICS*, pages 279–290, 2001.
4. N. Bansal, T Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54, 2007.
5. D. R. Cox. A use of complex probabilities in the theory of stochastic processes. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(2):313–319, 1955.
6. M. Elahi and C. Williamson. Autoscaling effects in speed scaling systems. In *Proc. of IEEE MASCOTS*, pages 307–312, 2016.
7. J. George and J. Harrison. Dynamic control of a queue with adjustable service rate. *Operations Research*, 49(5):720–731, 2001.
8. M. Harchol-Balter, N. Bansal, B. Shroeder, and M. Agrawal. Implementation of SRPT scheduling in web servers. In *Proc. of IEEE International Parallel & Distributed Processing Symposium (IPDS)*.

9. L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications.*
10. A. G. Pakes. Some conditions for ergodicity and recurrence of Markov chains. *Operations Research*, 17(6):1058–1061, 1969.
11. I. A. Rai, G. Urvoy-Keller, M. K. Vernon, and E. W. Biersack. Performance analysis of las-based scheduling disciplines in a packet switched network. In *Proc. of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '04/Performance '04)*, pages 106–117, 2004.
12. L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678–690, 1968.
13. A. Skrenes and C. Williamson. Experimental calibration and validation of a speed scaling simulator. In *Proc. of IEEE MASCOTS*, pages 105–114, 2016.
14. A. Wierman. Fairness and scheduling in single server queues. *Surveys in Operations Research and Management Science*, 6(1):39–48, 2011.
15. A. Wierman, L. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *Proc. of IEEE INFOCOM*, 2009.
16. A. Wierman, L. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Performance Evaluation*, 69:601–622, 2012.
17. F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proc. of ACM Foundations of Computer Systems (FOCS)*, pages 374–382, 1995.