CWS: a Model-Driven Scheduling Policy for Correlated Workloads

Giuliano Casale
Department of Computing
Imperial College London
180 Queen's Gate
London SW7 2AZ
g.casale@imperial.ac.uk

Ningfang Mi
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA, 02115
ningfang@ece.neu.edu

Evgenia Smirni
Department of Computer Science
College of William and Mary
Williamsburg, VA, 23187
esmirni@cs.wm.edu

ABSTRACT

We define CWS, a non-preemptive scheduling policy for workloads with correlated job sizes. CWS tackles the scheduling problem by inferring the expected sizes of upcoming jobs based on the structure of correlations and on the outcome of past scheduling decisions. Size prediction is achieved using a class of Hidden Markov Models (HMM) with continuous observation densities that describe job sizes. We show how the forward-backward algorithm of HMMs applies effectively in scheduling applications and how it can be used to derive closed-form expressions for size prediction. This is particularly simple to implement in the case of observation densities that are phase-type (PH-type) distributed, where existing fitting methods for Markovian point processes may also simplify the parameterization of the HMM workload model.

Based on the job size predictions, CWS emulates size-based policies which favor short jobs, with accuracy depending mainly on the HMM used to parametrize the scheduling algorithm. Extensive simulation and analysis illustrate that CWS is competitive with policies that assume exact information about the workload.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Sequencing and Scheduling; C.4 [**Performance of Systems**]: Performance Attributes

General Terms

Algorithms, Performance, Theory

Keywords

Stochastic scheduling, response time, model-driven scheduling, correlated workload

1. INTRODUCTION

We introduce CWS a general-purpose non-preemptive correlated workload scheduler. CWS uses the correlation structure of the workload and the observed sizes of past completed tasks to emulate size-based policies when only statistical information is available on upcoming jobs and on their relative order. This policy is motivated by the presence in system workloads of correlations that can strongly affect performance [7,18]. Scheduling such workloads is challenging because it is not easy to develop policies that leverage on all the available information from past scheduling decisions in an accurate and computationally efficient manner, while simultaneously estimating and leveraging the correlation structure of the workload. This paper offers an approach to tackle this problem based on Hidden Markov Model (HMM) theory [17]. CWS emulates scheduling policies that favor short jobs [22] based on closed-form expressions of expected job sizes that account for correlations. These expressions are obtained using an approach similar to the forward-backward algorithm of HMMs [17], which is used to determine the probability of a particular observation sequence in a HMM, and *exactly* account for the entire statistical characterization of the workload that can be inferred from the input HMM. As a result, CWS is able to take the best-informed decisions with respect to the available statistical model that describes the workload.

In general, existing scheduling techniques can be classified according to the assumed knowledge on the size of jobs. In *size-based* policies, such as SRPT [22] and its adaptive versions [12], the size of each job is known in advance and the scheduler takes advantage of this information to prioritize short jobs. On the other hand, *size-blind* policies, such as FB or PBS [8], do not rely on this information and thus can be easier to implement in systems where processing times are hard or impossible to determine in advance. Policies such as SEPT and its weighted version WSEPT [15, 20] know a priori only the *class* of each job in the queue. Each class has known size expectation and the policy prioritizes the oldest arrived job with smallest expected size. Conversely, methods such as FCFS, LCFS, and the random policy RAND completely ignore any information about job sizes or size class memberships. CWS differs from these policies as it infers knowledge of size class memberships based on the outcome of past scheduling decisions and on a HMM model that approximate the stochastic behavior of the workload. Although based on estimates, CWS greatly improves response times compared to policies such as FCFS or LCFS and often approaches the performance of SEPT that has exact information about the workload and thus provides a bound on the best performance achievable by CWS.

Summarizing, CWS uses the workload model to make scheduling decisions and we show by simulations that the resulting performance is highly competitive with that of SEPT which relies on exact knowledge of job class memberships. CWS instead es-

SIGMETRICS'10, June 14–18, 2010, New York, New York, USA.

timates the job class membership using the workload model and conditional on the outcomes of past scheduling decisions.

The paper is organized as follows. Problem statement is given in Section 2. The CWS algorithm is given in Section 3. Analysis of limiting cases is given in Section 4, while results of extensive simulation experiments are presented in Section 5 showing CWS scheduling performance under different workloads. Related work is summarized in Section 6. Finally, Section 7 concludes the paper with remarks on future work.

2. PROBLEM STATEMENT

We consider the problem of scheduling the execution of a fixed set of n jobs having service times represented by the random variables $S_1, \ldots, S_n$, where $S_j > 0, 1 \leq j \leq n$. The sequence of jobs $S_1, \ldots, S_n$ to be scheduled is referred to as the system *workload*. We assume in the following sections that the workload has fixed length n, meaning that the system does not accept new arrivals; we discuss the implementation and performance of CWS in presence of an exogenous arrival process in Section 5. Without loss of generality, S_j may equivalently be seen as the size of the jth job. Throughout the paper we use the terms service time and job size interchangeably. We take very broad assumptions about the system:

(A1) the exact size of a job is known only when the job completes execution (*size-blind scheduling*);

(A2) the job in execution is served without interruptions until completion (*non-preemptive scheduling*);

(A3) the server cannot become idle if there are jobs left to be processed (*work-conserving scheduling*);

(A4) statistical correlations exist between job sizes

The target of our investigation in the next sections is to define the CWS scheduling policy in such a way that its mean response time $E[R_{\mathsf{CWS}}]$ improves over the mean response times of similar non-preemptive policies in presence of correlated workloads.

3. ALGORITHM

CWS is a model-based policy in which the scheduler uses a continuous-time HMM workload model $\mathscr{W}$ to summarize the properties of past-observed job sizes and to predict sizes of jobs that have not already been served. The static or dynamic fitting of such model from historical data is out of the scope of present paper, where we assume ideal conditions where $\mathscr{W}$ is always representative of the real workload. The hidden states of this HMM describe the job size classes. We consider an offline parameterization, where the model $\mathscr{W}$ is assumed known and remains constant over time. The scheduler treats the current workload as a sample path of $\mathscr{W}$ initialized in stationary state. Resorting to an offline parameterization means, in practice, that the model $\mathscr{W}$ is always assumed representative of the workload $S_1, \ldots, S_n$ although this is not used to parametrize $\mathscr{W}$.

The remainder of this section is as follows. In Section 3.1, we define the workload model $\mathscr{W}$ used by CWS. Section 3.2 explains how CWS performs a scheduling decision based on $\mathscr{W}$. Section 3.3 gives an algorithmic definition of CWS and discusses computational costs.

3.1 Workload Model

We describe correlated workloads using a HMM with the following characteristics. We assume that the n job sizes are observations

Table 1: Summary of workload model notation

$\mathscr{W}$	workload model
S_j	service time of jth job in queue (random variable)
t_j	service time of jth job in queue (observed)
F_k	cumulative distribution function of class-k job sizes
$\mathsf{X}(j)$	index of jth job scheduled under policy X
n	number of jobs composing the workload
m	index of current scheduling decision, $m = 1, \ldots, n$
$R_{\mathsf{X}}(n)$	response time of X for a given workload of length n
$E[S]$	mean job size
$E[S_j \vert m]$	expected size of jth job at the time of the mth decision
ε_{X}	mean number of scheduling errors under policy X
$\mathbb{E}[R_{\mathsf{X}}]$	mean $R_{\mathsf{X}}(n)$ under random workloads S of length n
$\boldsymbol{P}$	DTMC embedded at jump times in $\mathscr{W}$
$\boldsymbol{H}_j$	history matrix of job j
$\boldsymbol{l}_j$	history vector of jobs before position j
$\boldsymbol{r}_j$	history vector of jobs after position j
$p_{k,c}$	prob. that a job of class k is followed by a job of class c
$\boldsymbol{M}_i$	ith moment matrix of $\mathscr{W}$
$\boldsymbol{\pi}$	steady-state class membership row vector
$\boldsymbol{Q}(t)$	semi-Markov kernel
$\mathbf{1}$	column vector of all ones
σ	correlation decay rate in two state workload models

of a HMM $\mathscr{W}$ with K hidden states, each modeling a workload class. Each class $k = 1, \ldots, K$, represents a group of jobs with sizes that come from the same probability distribution. Throughout this paper, differently from the most diffused classes of HMMs, we focus on the case of general continuous observation densities for each state that are used to describe a job size. With this assumption, a class k is defined by two parameters:

- the observation density function, i.e., a cumulative distribution function $F_k \equiv F_k(t)$ used to sample the size t of class-k jobs, with the additional assumption that $F_k(t)$ has no probability mass at $t \leq 0$. The sample job size is the observation associated to the hidden state k.

- a probability vector $(p_{k,1}, p_{k,2}, \ldots, p_{k,K})$, which characterizes job size correlations. The vector describes the probability $p_{k,c}$ that, if the job size S_j has been sampled from class k, then the job size S_{j+1} will be sampled from class c.

A graphical explanation of the HMM workload model $\mathscr{W}$ is given in Figure 1. The figure depicts a model with $K = 3$ classes where job sizes are currently sampled from class 1. Note that the states are hidden since one can only observe the size of the observed jobs, but not the class of membership. Further, jobs of different classes may be sampled with similar sizes, thus making inference of the class membership for a given job difficult. We consider the model initialized using the state-steady class membership probabilities; after generating n samples, the model has created a sample workload.

3.1.1 Equivalent Semi-Markov Process Description

Let X_j denote the class of the jth job and S_j be its size. According to the above definitions, the HMM description may be seen as equivalent to modeling the workload as a sequence of holding times $h_1, \ldots, h_n$ that occur between n consecutive state jumps, such that $S_j = h_j$, $j = 1, \ldots, n$. The discrete-time Markov chain (DTMC) embedded at jump times in this equivalent semi-Markov process is

$$\boldsymbol{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,K} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ p_{K,1} & p_{K,2} & \cdots & p_{K,K} \end{bmatrix},$$

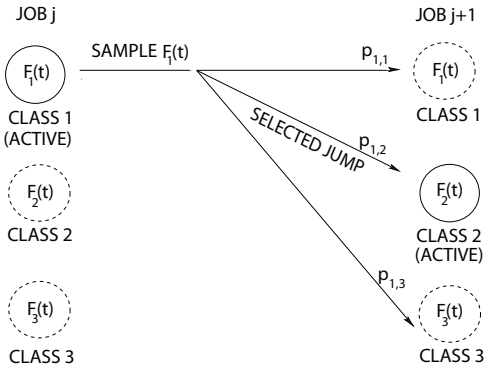

Figure 1: Sampling job sizes from a workload model $\mathscr{W}$ with $K = 3$ classes. The jth jump provides an estimate of the size of the jth job.

$\boldsymbol{P}\mathbf{1} = \mathbf{1}$, $\mathbf{1} = (1, 1, \ldots, 1)^T$, and it describes the correlations between job sizes. We denote by $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)$, $\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}$, the steady-state equilibrium vector of $\boldsymbol{P}$. The term π_k represents the equilibrium probability that the next job is member of class k when we look at the process immediately after the generation of a sample; we call $\boldsymbol{\pi}$ the steady-state class membership vector.

According to the above definitions, the HMM $\mathscr{W}$ may be seen as a semi-Markov process with kernel

$$\boldsymbol{Q}(t) = diag(F_1(t), F_2(t), \ldots, F_K(t))\boldsymbol{P} = \begin{bmatrix} F_1(t)p_{1,1} & F_1(t)p_{1,2} & \ldots & F_1(t)p_{1,K} \\ F_2(t)p_{2,1} & F_2(t)p_{2,2} & \ldots & F_2(t)p_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ F_K(t)p_{K,1} & F_K(t)p_{K,2} & \ldots & F_K(t)p_{K,K} \end{bmatrix}, \quad (1)$$

where the element (i, j) of the kernel describes the conditional probability that a job of size t and class i will be followed by a job of class j. Note that holding times are independent of the destination state, thus this is a special type of semi-Markov process.

Finally, we define the moment matrix

$$\boldsymbol{M_1} = \int_0^\infty t d\boldsymbol{Q}(t) = \begin{bmatrix} M_1p_{1,1} & M_1p_{1,2} & \ldots & M_1p_{1,K} \\ M_2p_{2,1} & M_2p_{2,2} & \ldots & M_2p_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ M_Kp_{K,1} & M_Kp_{K,2} & \ldots & M_Kp_{K,K} \end{bmatrix},$$

where the terms M_k, $1 \leq k \leq K$, denote the mean of the class-k job sizes. According to this definition, the mean job size $E[S]$ in the workload model $\mathscr{W}$ is given immediately by $E[S] = \boldsymbol{\pi}\boldsymbol{M_1}\mathbf{1}$.

3.1.2 Workload Model Generality

The HMM $\mathscr{W}$ is quite general since in the case where $K = n$ and each class has a deterministic distribution one may even describe exactly a given sequence of n job sizes. The most significant assumption behind $\mathscr{W}$ is the Markovian transition probability matrix $\boldsymbol{P}$ which limits the correlations between job sizes to the short-range dependent (SRD) type [24]. This happens because the correlations must decay geometrically as the powers of the eigenvalues of $\boldsymbol{P}$ different from 1. In the rest of the paper, these eigenvalues are referred to as correlation decay rates. Despite the geometric decay, it is established that SRD processes with many states can approximate well the performance of real workloads of long-range dependent (LRD) type [1, 6]. Thus, the Markovian nature of the state jumps does not appear very restrictive.

3.2 Scheduling Decisions

In this subsection, we describe how CWS uses effectively the workload model $\mathscr{W}$ for scheduling decisions [1]. We begin by considering the mth scheduling decision. CWS schedules for execution, among the $n - m + 1$ jobs left to be served, the job j with minimum expected size. This is done in a way that accounts for past scheduling history as explained in the following definition.

DEFINITION 1 (CWS SCHEDULING). *At the time of the mth decision, the CWS scheduling policy selects job j if and only if job j minimizes the expectation*

$$E[S_j|m] \equiv E[S_j|S_{\mathsf{CWS}(1)}, \ldots, S_{\mathsf{CWS}(m-1)}] \quad (2)$$

among the $n - m + 1$ jobs left to be served and conditional on the past decisions $\mathsf{CWS}(1), \ldots, \mathsf{CWS}(m-1)$, where $\mathsf{CWS}(i) = k$ denotes that job k has been selected by CWS at the time of the ith decision. In case of tie, CWS selects the job with minimum index j among the jobs with minimal value of $E[S_j|m]$[2].

The fundamental difference of CWS with respect to policies with exact class information like SEPT [20] is that CWS leverages on past scheduling decisions to select the next job to serve and this is done even if the class of membership of each job is unknown, i.e., the HMM hidden state that generated a given job is not known exactly.

To understand this concept, we first briefly summarize SEPT. For each of the n jobs, SEPT knows deterministically their class of membership, i.e., it knows that the job j is of class k with a job size that is distributed according to a given distribution F_k. Then the job having the minimum expected size is selected for execution. For instance, SEPT may know that job j is exponentially distributed with rate λ_k, while job l is exponentially distributed with rate $\lambda_h < \lambda_k$; in this case, job j is selected for execution because its expected size is less than for job l. Clearly, this is less accurate than size-based methods like SRPT or SJF , which know exactly the size of each job. The source of SEPT inaccuracies are essentially that (a) jobs of the same classes are indistinguishable and thus scheduled in FCFS order; (b) because of randomness, there is always some probability that a job belonging to a class with large mean is sampled small and vice versa.

Although SEPT can take erroneous decisions compared to size-based policies, its assumptions still provide an oversimplification with respect to the problem considered in this paper: in absence of externally provided information on the workload, only the correlation structure of job sizes can be estimated and used by CWS to estimate if job j may belong to class k. This implies a much harder analysis than in SEPT, since CWS must evaluate and update efficiently the conditional expectations $E[S_j|m]$ during the scheduling process. The fundamental property of the CWS workload model $\mathscr{W}$ presented in the previous section is that we are able to compute $E[S_j|m]$ analytically without the need of numerical integration. This is important because the existence of correlations between job sizes would in general require the computation of several nested integrals to determine $E[S_j|m]$. For example, assume that job $j = 1$

[1] We remark that, although our Markovian model may appear similar to a Markov decision process (MDP), this is not the case because we consider hidden states and the current state must be estimated from the observed job sizes. This type of decision processes belongs to the class of partially observable MDPs (POMDPs) which are known to be computationally intractable in most cases [14].

[2] This is an arbitrary decision in the case of a fixed-length workload. However, in the case of a system with exogenous arrivals treated in Section 5, this is desirable since it is the logical choice of serving the "oldest" job.

is served first; then computing the expected size $E[S_3|2]$ of job $j = 3$ at the time of the second scheduling decision requires to evaluate the $n-1$ dimensional integral

$$\int t_3 \cdot \mathbb{P}[\, S_2 = t_2,\, S_3 = t_3, \ldots, S_n = t_n \,|S_1]\, dt_2 \cdots dt_n,$$

over the region $t_2 \geq 0 \wedge t_3 \geq 0 \wedge \cdots \wedge t_n \geq 0$. In general, this is impractical because, if the workload is correlated, the conditional joint probabilities do not factor into a product-form, i.e.,

$$\mathbb{P}[\, S_2 = t_2,\, S_3 = t_3, \ldots, S_n = t_n \,|S_1] \neq \prod_{i=2}^{n} \mathbb{P}[S_i = t_i|S_1],$$

and thus the integral solution is usually not available in closed analytical form. Multi-dimensional numerical integration is clearly too expensive to be used in a scheduling algorithm. The HMM workload model $\mathscr{W}$ has instead the key property of removing this analytical intractability by a closed-form expression for the joint probabilities which avoids numerical integration. This stems from the analytical tractability of the considered class of HMMs.

3.2.1 Forward-Backward Algorithm

We show how job sizes are estimated from $\mathscr{W}$ without resorting to numerical integration and based on the forward-backward algorithm of HMMs [17]. Using the semi-Markov interpretation of the HMM model given in Section 3.1.1, we define the *history matrix* of job i as

$$\boldsymbol{H}_i(m) = \begin{cases} \boldsymbol{P} & \text{if } \nexists\, m' < m :\ \mathsf{CWS}(m') = i; \\ \boldsymbol{C}(t_i) & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, n$, where

$$\boldsymbol{C}(t_i) = \frac{d\boldsymbol{Q}(t)}{dt}\Big|_{t=t_i}, \tag{3}$$

and t_i is the observed size of job i, which is $\boldsymbol{H}_i(m) = \boldsymbol{P}$ if and only if job i has not been scheduled before the mth decision of CWS. The element (i, j) of $\boldsymbol{P}$ gives the conditional probability that, if i is the hidden state which generated the size of job m, then j will be the state generating the size of job $m+1$. The corresponding element of $\boldsymbol{C}(t_i)$ accounts also for the condition that the ith job size has been observed equal to t_i.

From this, we can apply the forward-backward algorithm of HMMs for computing the conditional expected job sizes in the workload model $\mathscr{W}$ as follows.

PROPOSITION 1 (JOB SIZE EXPECTATION). *If the workload is characterized by the HMM $\mathscr{W}$, the expected size of job j at the instant of the m-th scheduling decision is*

$$E[S_j|m] = \frac{\boldsymbol{\pi} \prod_{i=1}^{j-1} \boldsymbol{H}_i(m) \boldsymbol{M_1} \prod_{k=j+1}^{n} \boldsymbol{H}_k(m) \mathbf{1}}{\boldsymbol{\pi} \prod_{i=1}^{j-1} \boldsymbol{H}_i(m) \boldsymbol{P} \prod_{k=j+1}^{n} \boldsymbol{H}_k(m) \mathbf{1}}, \tag{4}$$

for all jobs $j = 1, \ldots, n$ and scheduling decisions $m = 1, \ldots, n$.

PROOF. We derive the expected job size $E[S_j|m]$ by first formulating the conditional probability $\mathbb{P}[S_j = t\,|m]$ and then obtaining the closed-form formula for

$$E[S_j|m] = \int_0^{\infty} t\mathbb{P}[S_j = t\,|m] dt.$$

We begin by rewriting the conditional probability $\mathbb{P}[S_j = t\,|m]$ as

$$\frac{\mathbb{P}[S_j = t, S_{\mathsf{CWS}(1)} = t_{\mathsf{CWS}(1)}, \ldots, S_{\mathsf{CWS}(m-1)} = t_{\mathsf{CWS}(m-1)}]}{\mathbb{P}[S_{\mathsf{CWS}(1)} = t_{\mathsf{CWS}(1)}, \ldots, S_{\mathsf{CWS}(m-1)} = t_{\mathsf{CWS}(m-1)}]}.$$

This last expression is a ratio of joint probabilities which can be evaluated efficiently using the workload model $\mathscr{W}$. According to these observation and using standard arguments of the forward-backward algorithm we conclude that the joint probability is computed from (3.2.1) as

$$\frac{\boldsymbol{\pi} \prod_{i=1}^{j-1} \boldsymbol{H}_i(m) \boldsymbol{C}(t) \prod_{i=j+1}^{n} \boldsymbol{H}_i(m) \mathbf{1}}{\boldsymbol{\pi} \prod_{i=1}^{j-1} \boldsymbol{H}_i(m) \boldsymbol{P} \prod_{i=j+1}^{n} \boldsymbol{H}_i(m) \mathbf{1}},$$

where the denominator normalizes over the probability of observing exactly the job sizes found in the first $m-1$ served jobs (job j has here the matrix $\boldsymbol{P}$ since it has not been served among the first $m-1$ jobs and the probability space is at the time of the mth decision). The final result follows by evaluating

$$\int_0^{+\infty} t\mathbb{P}[S_j = t\,|m] dt$$

which gives the final expression for $E[S_j|m]$ by noting that

$$\int_0^{+\infty} t\boldsymbol{C}(t) dt \quad = \quad \int_0^{+\infty} t \left(\frac{d\boldsymbol{Q}(t)}{dt} \right) dt \quad = \quad \boldsymbol{M_1}.$$

□

The formula obtained in Proposition 1 is an expression for the forward-backward algorithm for the workload model $\mathscr{W}$ and it is the fundamental tool for evaluating the expected job sizes in correlated workloads. The expression is exact also if job sizes are uncorrelated: in this case, it is not difficult to see by spectral decomposition [19] that $\boldsymbol{P} = \mathbf{1}\boldsymbol{\pi}$ and this correctly transforms (4) into a product-form expression.

3.2.2 PH-type Job Sizes and HMM Fitting

A case of major interest in performance evaluation is when job sizes are described by means of a PH-type distribution. As we show below, this brings major benefits in terms of compactness of the representation and provides a measurement-driven fitting approach for the definition of HMM model from past observations.

It is interesting to note that if all distributions $F_k(t)$ are PH-type, then the HMM model presented is equivalent to Neuts' Markovian Arrival Process (MAP) with J states and representation $(\boldsymbol{D_0}, \boldsymbol{D_1})$, where $\boldsymbol{P} = (-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}$. This equivalence can be easily verified at the semi-Markov level, see [10] for an introduction. The matrix $\boldsymbol{D_1}$ is square of order J and represents rates of transitions that lead to absorption events; conversely, $\boldsymbol{D_1}$ describes non-absorbing transitions and has negative diagonal elements such that $\boldsymbol{D_0} + \boldsymbol{D_1}$ is the infinitesimal generator of the underlying CTMC that modulates the active state.

Formulating the HMM as a MAP(J) has the advantage that one can use a Markov model with a space of $J < K$ states to jointly describe all PH-type probability distributions $F_k(t)$, whereas in the general case discussed so far one may need to maintain a detailed description of each of the K distributions $F_k(t)$. This is because one can associate class membership to activation of a certain transition of $\boldsymbol{D_1}$, thus the number of classes that can be represented with a MAP grows quadratically with the state space size J. Even more importantly, a MAP description allows to use existing fitting techniques for MAPs to automatically derive the workload model [1]. This greatly widens the applicability of CWS by providing a way to use existing results for Markovian workload fitting to define the workload model $\mathscr{W}$.

Within the MAP modeling description, we may reformulate the forward-backward formula for job size expectations (4) as follows. We have that the first moment matrix of a MAP is $\boldsymbol{M_1} = (-\boldsymbol{D_0})^{-1}$

and we observe that the definition of history matrix immediately implies that

$$\boldsymbol{H}_i(m) = \begin{cases} (-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}, & \text{if } \nexists\, m' < m:\ \mathsf{CWS}(m') = i; \\ \mathbf{e}^{\boldsymbol{D_0}t_i}\boldsymbol{D_1}, & \text{otherwise,} \end{cases}$$

that is equivalent to[3]

$$\boldsymbol{P} = (-\boldsymbol{D_0})^{-1}\boldsymbol{D_1} \tag{5}$$

$$\boldsymbol{C}(t_i) = \mathbf{e}^{\boldsymbol{D_0}t_i}\boldsymbol{D_1} \tag{6}$$

where the matrix exponential function is[4]

$$\mathbf{e}^{\boldsymbol{D_0}t_i} = \sum_{k=0}^{\infty} \frac{(\boldsymbol{D_0}t_i)^k}{k!} \tag{7}$$

The above expression allows a reformulation of (4) using an efficient implementation based on MAPs in the case of continuous distributions with unbounded support. To the best of our knowledge, such a case has been tackled in HMMs for Gaussian and log-concave distributions [17], but has not been addressed for general PH-type distributions.

3.3 CWS Scheduling Algorithm

As observed in Definition 1, for each scheduling decision CWS selects the job with the minimum expected size as inferred from correlations and observed sizes of past served jobs. This expectation is computed by CWS using $\mathscr{W}$ and the formula in Proposition 1. After completing the newly scheduled job j, the history matrix $\boldsymbol{H}_j(m)$ is changed from $\boldsymbol{P}$ to $\boldsymbol{C}(t_j)$. Thus the next scheduling decision is performed in a similar manner, but with different conditional job size expectations because of the change in $\boldsymbol{H}_j(m)$. When all n jobs have been served the algorithm completes execution. Details about the implementation of CWS are discussed below.

For each of the n scheduling decisions, computing the expected job sizes $E[S_j|m]$ for a workload of size n with K classes has a computational requirement of $O(nK^2)$ operations due to the matrix products and $O(nK^2)$ space due to storing the $\boldsymbol{H}_j(m)$ matrices in main memory. Thus the overall cost of scheduling is $O(n^2K^2)$ in time and $O(nK^2)$ in space. However, a more memory-efficient implementation of (4) can be obtained by storing n vectors $\mathbf{l}_j$ and $\mathbf{r}_j$ in place of the $\boldsymbol{H}_j(n)$ matrices. Such vectors summarize the history of jobs with indexes before and after job j. These vectors have length K and are given by

$$\mathbf{l}_j = \boldsymbol{\pi}\prod_{i=1}^{j-1}\boldsymbol{H}_i(m), \quad \mathbf{r}_j = \prod_{i=j+1}^{n}\boldsymbol{H}_i(m)\mathbf{1},$$

for all $j = 1, \ldots, n$. It is immediate to see from Proposition 1 that

$$E[S_j|m] = \frac{\mathbf{l}_j\boldsymbol{M_1}\mathbf{r}_j}{\mathbf{l}_j\boldsymbol{P}\mathbf{r}_j}. \tag{8}$$

Note also that both vectors can be computed on-the-fly without the need of storing the $\boldsymbol{H}_j(m)$ matrices.

A pseudo-code summarizing how CWS scheduling works based on the vectors $\mathbf{l}_j$ and $\mathbf{r}_j$ is reported in Figure 2. Note that only the vectors $\mathbf{l}_j$ and $\mathbf{r}_j$ affected by the last served job are updated in the last two **forall** cycles. In our notation, if $\mathsf{CWS}(m) + 1 > n$ then the cycle for $j = \mathsf{CWS}(m) + 1, \mathsf{CWS}(m) + 2, \ldots, n$ is skipped, the case where $\mathsf{CWS}(m) < 2$ in the other cycle is treated similarly. Note also that, for large K, storing the vectors $\mathbf{l}_j$ and $\mathbf{r}_j$ instead of the $\boldsymbol{H}_j(m)$ matrices provides substantial memory savings since the storage complexity grows in this way only as $O(nK)$. We also remark that, for large n and depending on the numerical values of the matrices, numerical scaling may be needed to ensure that (8) does not suffer floating-point range underflows or overflows. The scaling factors used should be taken into account when comparing the expected sizes of jobs. Additionally, it is often useful to ignore the denominator of (8) that is identical for all values of j, thus it does not really affect the CWS decision.

Within a queueing environment, the implementation of CWS is slightly different from the one described above because of the presence of an arrival process. In this case, the number of jobs n in the system varies over time. CWS uses an estimation window of W jobs such that only the expected sizes of unserved jobs between position j_0 and $j_0 + W - 1$ are considered for scheduling, where j_0 is the index of the oldest unserved job in the system. If all jobs in position $i < j_0$ have been served, CWS computes and keeps in memory the membership vector $\boldsymbol{\pi}_{j_0}$ for position j_0, which is used in place of the vector $\boldsymbol{\pi}$ in the formula of Proposition 1. The history matrices before position j_0 are all discarded since $\boldsymbol{\pi}_{j_0}$ completely summarizes past system history. When the oldest unserved job in the system is scheduled for service, j_0 is updated to the position of the second-oldest unserved job and thus the window slides forward.

There are several reasons for adopting a sliding window. On the one hand, it seems reasonable that practical implementations of CWS would need to limit the policy's computational effort: a window immediately controls the number of times the conditional expectation in Proposition 1 is evaluated and bounds the maximum memory occupation. Another reason is that a sliding window W reduces the maximum starvation of large jobs in CWS. Finally, when the distance between the occurrences of jobs increases beyond a certain point, the correlation between job sizes becomes infinitesimal, so job sizes may be treated as independent for any practical use.

3.4 Illustrative Example

We illustrate the CWS scheduling algorithm using a case study. We consider a synthetic workload consisting of 65 jobs with sizes randomly generated using the model $\mathscr{W}$, see Figure 4(a). The workload model consists of $K = 3$ job size classes with transition probabilities

$$\boldsymbol{P} = \begin{pmatrix} 0.90 & 0.10 & 0 \\ 0 & 0.80 & 0.20 \\ 0.10 & 0 & 0.90 \end{pmatrix}.$$

For simplicity, the job size distributions are all deterministic such that the job sizes are $S_j = 0.5$ for jobs of class 1, $S_j = 1$ for class 2, and $S_j = 2$ for jobs of class 3. According to these values and the transition probabilities in $\boldsymbol{P}$, the workload model $\mathscr{W}$ has jobs sizes with mean $E[S] = 1.2$ and coefficient-of-variation $CV = 1.28$. Because of the randomness of the sampling process, some discrepancy exists between the sample workload and the workload model $\mathscr{W}$: the empirical class membership probabilities for a job are $(0.35, 0.40, 0.25)$ against the stationary value $(0.40, 0.20, 0.40)$ of $\mathscr{W}$. As we show in the example, since CWS cannot anticipate the empirical distribution, it is slightly more aggressive than needed because it expects in the workload 0.40 percent of class 3 jobs which have large size. As a result, it makes some precautionary scheduling decisions that differ from a FCFS rule in order to avoid serving the large jobs. We henceforth refer to these changes of scheduling order with respect to FCFS as "jumps". Concerning

[3] Note that, if $F_k(t)$ is not exponential, $\boldsymbol{P}$ and $\boldsymbol{C}(t_i)$ account also for the different phases that specify the $F_k(t)$ distribution.

[4] The matrix exponential is implemented efficiently in several software packages, e.g., as the expm function in MATLAB. In ad-hoc implementations, it may be computed in several way, e.g., by truncation of (7) or by spectral methods using the eigenvalues and eigenvectors of $\boldsymbol{D_0}t_i$.

CWS **Scheduling Algorithm**
/ initialize */*
$\mathbf{l}_1 = \boldsymbol{\pi}$;
$\mathbf{r}_n = \mathbf{1}$;
forall jobs $j = 2, \ldots, n-1$
  $\mathbf{l}_j = \mathbf{l}_{j-1}\boldsymbol{P}$;
forall jobs $j = n, \ldots, 2$
  $\mathbf{r}_{j-1} = \boldsymbol{P}\mathbf{r}_j$;
/ start scheduling */*
forall scheduling decisions $m = 1, \ldots, n$
  / compute expected job sizes */*
  forall jobs $j = 1, \ldots, n$
    $E[S_j|m] = +\infty$;
    if job j has not been served
      compute $E[S_j|m]$ according to (8);
  / scheduling decision */*
  mth decision: $\mathsf{CWS}(m) = \min_j E[S_j|m]$
  serve job $\mathsf{CWS}(m)$ and call its size $t_{\mathsf{CWS}(m)}$;
  / update conditional prob. based on observed size */*
  forall jobs $j = \mathsf{CWS}(m)+1, \mathsf{CWS}(m)+2, \ldots, n$
    $\mathbf{l}_j = \mathbf{l}_{j-1}\boldsymbol{C}(t_{\mathsf{CWS}(m)})$;
  forall jobs $j = \mathsf{CWS}(m), \mathsf{CWS}(m)-1, \ldots, 2$
    $\mathbf{r}_{j-1} = \boldsymbol{C}(t_{\mathsf{CWS}(m)})\mathbf{r}_j$;

Figure 2: CWS **scheduling algorithm.**

the empirical transition probabilities, these are as follows

$$\widetilde{\boldsymbol{P}} = \begin{pmatrix} 0.86 & 0.14 & 0 \\ 0 & 0.88 & 0.12 \\ 0.25 & 0 & 0.75 \end{pmatrix},$$

which for the first two classes are close to the values in $\boldsymbol{P}$, while they have an error of about 15% on the transition probabilities of the third class. By showing that CWS performs well in this example, we provide intuition that it can leverage in an effective way the correlation information that is available from $\mathscr{W}$ and correlation estimates do *not* need to be extremely accurate to achieve good scheduling results.

The execution of the CWS algorithm is illustrated in Figure 3. The figure is a collection of four diagrams illustrating the value of the expected (*white*) and observed (*black*) job sizes as a function of the scheduling decision number $m \in \{3, 15, 30, 45\}$. The white bars represent size estimates of jobs that have still to be served, black bars summarize past history. The white bars are the values of the expected job sizes $E[S_j|m]$ at the time of the mth scheduling decision; the black bars, instead, are the exact job sizes observed after the job in the corresponding position has been served. Note that the heights of the white bars change over time because the estimate $E[S_j|m]$ is updated after each decision. Each diagram in Figure 3 includes a horizontal line, jobs bigger than this threshold belong all to the large class 3.

The top Figure 3(a) represents the first two scheduling decisions of CWS. At the time of the first scheduling decision, CWS has no specific information about the jobs in the system and thus decides to serve the first job. This is completed after $S_1 = 2$ units of time and hence it is recognized to be a long job of class 3. The second decision is to schedule the job $j = 19$. It can be shown that this position minimizes the expectation $E[S_j|S_1 = 2]$ since there is minimal correlation with the last served job that is large. Moreover, it is an effective decision because, from $\boldsymbol{P}$, the number of consecutive large jobs in $\mathscr{W}$ follows a geometric distribution with mean 9 and standard deviation about 9, thus after 18 jobs the

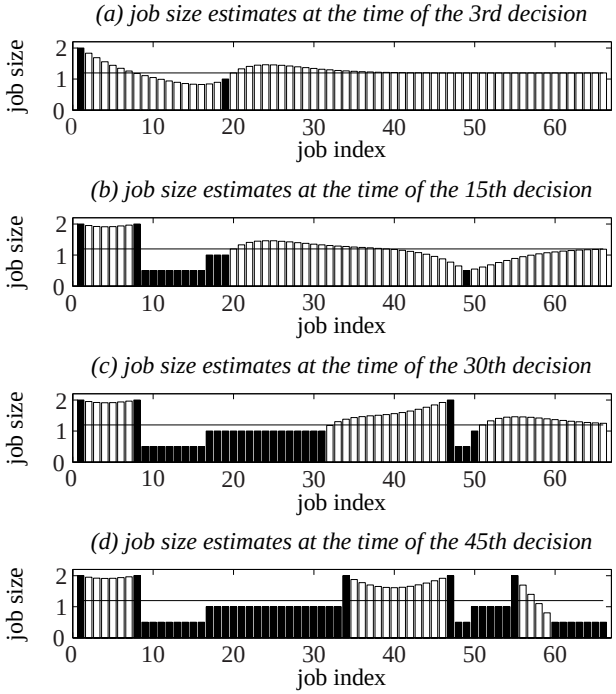

Figure 3: Intermediate steps of CWS **execution on the case study. Legend:** $\square$ **= expected job size (i.e.,** $E[S_j|m]$**),** $\blacksquare$ **= observed job size (i.e.,** S_j**),** $-$ **is the mean job size (i.e.,** $E[S] = 1.2$**).**

median behavior is to be out of the long burst. Note that the expectation $E[S_j|S_1 = 2]$ is not the one shown in Figure 3(a) which refers to the third decision, but is the expectation after the first decision, which is indeed different from Figure 3(a) because it is not yet known that the job $j = 19$ is small.

In Figure 3(b), after scheduling the job in position $j = 19$, this is found to be a small job and the algorithm performs the next 15 scheduling decisions by first exploring the area $j < 19$ surrounding the small job found and then jumping away when a large job is found in position $j = 8$. Here, the algorithm does not evaluate positions $j < 8$, which should contain large jobs, and decides not to explore further the surroundings of $j = 20$ which may contain a large job of class 3 too. In addition, the size minimization is more effective if class 1 jobs can be found rather then continuing to serve class 2 jobs near $j = 20$. Thus, a jump is done to $j = 49$ which again minimizes the job size expectation. Note that the jump is longer than in the early selection of the job $j = 19$ because a larger jump is needed to avoid the upcoming class 2 and class 3 jobs in the surroundings of $j = 20$, whereas for the initial jump to $j = 19$ only the class 3 jobs had to be avoided.

In Figure 3(c), CWS has performed fifteen additional decisions, which have involved initially the area around $j = 50$. When a large job is found in position $j = 47$, CWS jumps back to $j = 20$ and proceeds up to position $j = 31$. In Figure 3(d), after reaching position $j = 35$, another large job is found and CWS jumps back to the area of $j = 50$ from which, after finding at $j = 55$ another long job, it jumps again to the end of the sequence where a sequence of small jobs is served. Finally, the residual areas unexplored in Figure 3(d) are evaluated with several jumps. As we can see from Figure 4(a), these areas have the highest density of large jobs and it is a good outcome of the CWS scheduling that these areas are served last. It is also interesting to note that existing scheduling policies such as FCFS, LCFS, or RAND would have all failed in

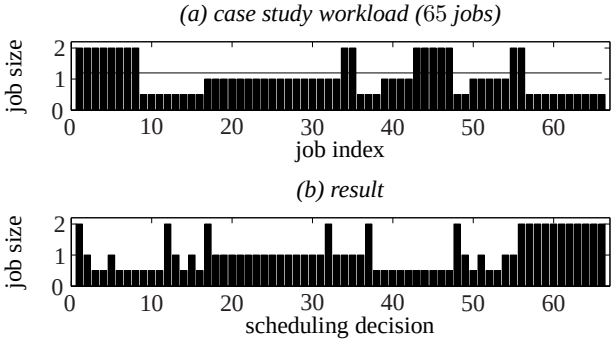

Figure 4: Case study of a workload with 65 jobs sampled from three classes with deterministic distribution and means 0.5, 1, and 2. The horizontal line in figure (a) is the mean job size $E[S] = 1.2$.

completing early the area $9 \leq j \leq 33$ which contains the largest amount of jobs with size less than the mean. Instead, CWS has been able to perform the useful first jump to the position $j = 19$ by evaluation of the transition probabilities $\boldsymbol{P}$ and, additionally, has leveraged this good position to serve the local group of small jobs.

The figures should be related to the two diagrams in Figure 4 which show the original job size sequence in Figure 4(a) and the same sequence reordered after CWS scheduling in Figure 4(b). Note immediately from Figure 4(b) that CWS is extremely effective, since most of the large jobs are served at the end of the sequence. It is also important to see that occasional mistakes, where a long job is scheduled before a small job, are not followed immediately by another error. Thus, the example illustrates the ability of CWS of detecting and serving small jobs in the workload.

4. LIMITING CASES

To investigate the behavior of CWS, we first consider a simple set of limiting cases that illustrate if CWS meets some basic features that are desirable under correlated workloads. In particular, we give a theoretical treatment, whereas in Section 5 we analyze CWS by simulations over a variety of workloads. Note that the limiting cases considered here are sufficiently simple to enable analytical tractability which is generally unfeasible for history-based methods, since the number of possible system histories that affect the present decisions grows exponentially with the workload size.

Methodology

For ease of interpretation, we consider workloads with jobs of two classes: small (s) and large (l). In addition, we assume that the two classes are fully disjoint, i.e., a job size t is uniquely mapped either to the small or to the large class; fully disjoint classes may be defined by cutting the cumulative distribution function of the measured job sizes in two halves and assigning each part to a different class. Given this constraint, the distribution function $F_k(t)$, $k \in \{s, l\}$, of each class can be arbitrary. We provide a characterization based on the following three limiting cases which illustrate desirable features of scheduling under correlated workloads.

Avoidance Test. The test considers a group of n_l consecutive large jobs inserted in a workload of $n_s = n - n_l$ small jobs. Thus this benchmark estimates if a scheduling policy can avoid serving large jobs when these appear in groups. The large jobs occupy the positions $l+1, l+2, \ldots, l+n_l$, where the *location parameter* l is a discrete uniform random variable taking values in $0, 1, \ldots, n_s$.

Seeking Test. The test is dual to the avoidance test: large jobs are replaced by small jobs and vice versa. Thus this benchmark estimates if a scheduling policy can find quickly groups of n_s contiguous small jobs immersed in a workload of $n_l = n - n_s$ large jobs. Note that for $n_s = n_l$ the seeking test is still different from the avoidance test because in the former there is always a single contiguous group of long jobs, while in the latter long jobs are in two groups spaced by the n_s short jobs.

Pattern Test. The benchmark evaluates how a policy leverages on patterns in the workload. The workload includes n_l large jobs which appear deterministically with frequency $1/T$. The first large job appears in position l equilikely in $0, \ldots, n_s$, where $n_s = n - n_l$ is the number of small jobs.

Average performance is evaluated using the mean number of errors ε and compared with the results of SEPT, FCFS, LCFS, and RAND. We point to Section 3.2 for a compact description of SEPT. The mean number of errors stands for the number of large jobs scheduled when one or more jobs belonging to the short class remain in the system and provides clear evidence of the accuracy of the scheduling in a way that is insensitive to the variability of the workload, thus it is representative of workloads with different levels of variability.

Avoidance Test

Consistently with the test assumptions, we consider for CWS a basic workload model $\mathscr{W}_2$ with two job size classes where

$$\boldsymbol{P} = \begin{bmatrix} p_s & 1-p_s \\ 1-p_l & p_l \end{bmatrix}, \boldsymbol{Q}(t) = diag(F_s(t), F_l(t))\boldsymbol{P}.$$

The distribution of the small and large states is fully disjoint and has means M_s and M_l, respectively. Since in the avoidance test long jobs appear in groups, we assume a parameterization of CWS with positive correlations between job sizes, which can be shown to be equal to imposing $p_l + p_s > 1$.

THEOREM 1. *In the avoidance test, for a workload of n jobs among which n_s small and n_l large,* CWS *makes on average*

$$\varepsilon_{\mathsf{CWS}} = \frac{n_s}{1+n_s} < 1, \tag{9}$$

errors, where the metric is averaged on all possible values of the location parameter l.

PROOF. We analyze the chain of decisions using spectral decomposition on the $\boldsymbol{P}$ matrix, see [19] for an overview. In the case of a model with two states, we recall that, $\boldsymbol{P}$ being a transition probability matrix, it is $\boldsymbol{P}^j = (1-\sigma^j)\mathbf{1}\boldsymbol{\pi} + \sigma^j \boldsymbol{I}$, where $\sigma = p_s + p_l - 1$ is the smallest eigenvalue of $\boldsymbol{P}$ and $\boldsymbol{I}$ is the identity matrix of order two. Since CWS has positive correlations, it is $0 < \sigma < 1$.

Analysis of decision $m = 1$. The expected job sizes are clearly $E[S_j|1] = E[S]$, for all indexes j.Thus, all jobs have initially the same expected size and CWS selects $j = 1$ by Definition 1.

Analysis of decision $m = 2$. We have

$$E[S_j|2] = \frac{\boldsymbol{\pi}_2 \boldsymbol{P}^{j-2} \boldsymbol{M_1} \boldsymbol{P}^{n-j} \mathbf{1}}{\boldsymbol{\pi}_2 \boldsymbol{P}^{n-1} \mathbf{1}},$$

where $\boldsymbol{\pi}_2 = \boldsymbol{\pi}\boldsymbol{C}(t_1)$.

$$\boldsymbol{\pi}_2 = \boldsymbol{\pi}\boldsymbol{C}(t_1) = \begin{bmatrix} \pi_s p_s & \pi_l(1-p_{sl}) \\ 0 & 0 \end{bmatrix}$$

if the first job is small and

$$\boldsymbol{\pi}_2 = \boldsymbol{\pi}\boldsymbol{C}(t_1) = \begin{bmatrix} 0 & 0 \\ \pi_l p_l & \pi_l(1-p_l) \end{bmatrix}$$

otherwise. By spectral decomposition on $\boldsymbol{P}^{j-2}$ it is

$$E[S_j|2] = \frac{\boldsymbol{\pi}_2(\mathbf{1}\boldsymbol{\pi} + \sigma^{j-2}(\boldsymbol{I} - \mathbf{1}\boldsymbol{\pi}))\boldsymbol{M_1}\mathbf{1}}{\boldsymbol{\pi}_2\mathbf{1}}$$
$$= (1-\sigma^{j-2})E[S] + \sigma^{j-2}\frac{\boldsymbol{\pi}_2\boldsymbol{M_1}\mathbf{1}}{\boldsymbol{\pi}_2\mathbf{1}},$$

where we used that $\boldsymbol{\pi}_2\mathbf{1} = 1$, $\boldsymbol{\pi}_2$ being a conditional distribution. Observing that $\boldsymbol{\pi}(1-\sigma) = ((1-p_l),(1-p_s))$, we conclude that if the size of the first job t_1 is small then

$$\frac{\boldsymbol{\pi}_2\boldsymbol{M_1}\mathbf{1}}{\boldsymbol{\pi}_2\mathbf{1}} = p_s M_s + (1-p_s)M_l,$$

which is less than $E[S]$ whenever $\sigma > 0$. Thus $E[S_j|2]$ is minimized for $j = 2$ which is the second decision. For t_1 large, instead,

$$\frac{\boldsymbol{\pi}_2\boldsymbol{M_1}\mathbf{1}}{\boldsymbol{\pi}_2\mathbf{1}} = (1-p_l)M_s + p_l M_l.$$

which is larger than $E[S]$ whenever $\sigma > 0$. Thus, if t_1 is large $E[S_j|2]$ is minimized for $j = n$ which is the second decision.

Analysis of the decisions $m = 3, \ldots, l+1$ when t_1 is small. The intermediate steps of the recursion proceed similarly since, after serving the mth small job, $m \leq l$, it is always the case that

$$E[S_j|m] = (1-\sigma^{j-m})E[S] + \sigma^{j-m}(p_s M_s + (1-p_s)M_l)$$

which makes $E[S_j|m]$ minimal by the choice $j = m$.

Analysis of the decision $m = l+2$ when t_1 is small. When CWS finds a large job in position $l+1$, the following decision leverages on the information carried by the error. In this case it is easy to verify that

$$\frac{\boldsymbol{\pi}_m\boldsymbol{M_1}\mathbf{1}}{\boldsymbol{\pi}_m\mathbf{1}} = \frac{\boldsymbol{\pi}_{m-1}\boldsymbol{C}(t_m)\boldsymbol{M_1}\mathbf{1}}{\boldsymbol{\pi}_{m-1}\boldsymbol{C}(t_m)\mathbf{1}} = (1-p_l)M_s + p_l M_l,$$

which makes $E[S_j|m]$ minimal by the choice $j = n$. Thus, CWS jumps to the farthest position $j = n$.

Analysis of the decisions $m = l+3, \ldots, n$ when t_1 is small. If $n - n_l - l \leq n_l$ then only large jobs are left to be served and the algorithm has served the workload without errors. Otherwise, another small job is found and we have the job size expectation

$$E[S_j|m] = \frac{\boldsymbol{\pi}\prod_{i=1}^{l}\boldsymbol{C}(t_i)\boldsymbol{C}(t_{l+1})\boldsymbol{P}^{j-l-2}\boldsymbol{M_1}\boldsymbol{P}^{n-j-1}\boldsymbol{C}(t_n)\mathbf{1}}{\boldsymbol{\pi}\prod_{i=1}^{l}\boldsymbol{C}(t_i)\boldsymbol{C}(t_2)\boldsymbol{P}^{n-l-2}\boldsymbol{C}(t_n)\mathbf{1}}$$

Using again spectral decomposition we get

$$E[S_j|m] = (1-\sigma^{j-l-2})(1-\sigma^{n-j-1})A + \sigma^{n-l-3}D$$
$$+ (1-\sigma^{j-l-2})\sigma^{n-j-1}B + \sigma^{j-l-2}(1-\sigma^{n-j-1})C,$$

where the terms A, B, C, and D are immediately given by the intermediate products of the matrices $\mathbf{1}\boldsymbol{\pi}$ and $\boldsymbol{I}$ in the spectral decomposition and from the expressions of $\boldsymbol{C}(\cdot)$, $\boldsymbol{M_1}$, and $\boldsymbol{P}$. In particular, it is easy to show from these expressions that $A > C > B \geq 0$. Let now $f(j)$ be the continuous version of $E[S_j|m]$ for real values of $j \in [l+2, n-1]$. Taking the second derivative it is found that this lies in

$$f''(j) \in [(A-B)f^*, (A-C)f^*],$$

where $f^* = -(\log\sigma)^2(\sigma^{n-j-1} + \sigma^{j-l-2})$, where both extremes are negative values if $0 < \sigma < 1$. Thus, $f''(j) < 0$ for all js and noting that on the extremes of the range of definition of $E[S_j|m]$ is

$$f(l+2) - f(n-1) = (1-\sigma^{n-l-3})(C-B) > 0$$

we conclude that $f(l+2) > f(n-1)$ and the minimum expected job size is located always on $j = n-1$; since this is an integer value, the same consideration applies also to $E[S_j|m]$: thus, the decision $m = l+3$ is to select the job $j = n-1$.

The same approach applies identically to the remaining decisions, since the above quantities A, B, C, and D are all subsequently scaled by a π_s constant after each scheduling decision and thus CWS always continues to serve the remaining job from the tail of the workload. This proves that the small jobs left are all served before the large ones and concludes the case where t_1 is small.

Analysis of the decisions $m = 3, \ldots, l+1$ when t_1 is large. The proof is identical to the case $m = l+2, \ldots, n$ when t_1 is small.

Average number of errors. According to the above observations, it is immediately found that the number of errors is 0 if $l = n_s$, which has probability $(1+n_s)^{-1}$, and 1 otherwise; this gives the expectation $\varepsilon_{\mathsf{CWS}} = n_s/(1+n_s)$. □

Avoid.		SEPT		CWS		FCFS/LCFS		RAND	
n	n_l	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$
10	*2*	*0.0*	35	*0.9*	75	*1.8*	114	*1.8*	115
10	*5*	*0.0*	154	*0.8*	179	*4.2*	278	*4.3*	278
10	*8*	*0.0*	362	*0.7*	372	*5.3*	441	*3.0*	440
100	*20*	*0.0*	258	*1.0*	298	*19.8*	1050	*19.7*	1050
100	*50*	*0.0*	1313	*1.0*	1338	*49.0*	2550	*49.1*	2550
100	*80*	*0.0*	3258	*1.0*	3268	*76.2*	4050	*76.9*	4050

Table 2: Avoidance test results for different values of the workload length n and of the number of large jobs n_l. The number of errors ε and the mean response time $\mathbb{E}[R]$ are averaged on different values of the location parameter l which are assumed equilikely. In the experiments, the mean value of job sizes are $M_s = 1$ for small jobs and $M_l = 100$ for large jobs and class distributions are deterministic. RAND **results are averaged over one thousand experiments.**

A quantitative example of the avoidance test is given in Table 2 on models with $M_s = 1$, $M_l = 100$, and workload length $n = 10, 100$. Results are obtained by running the different algorithms on the avoidance test workloads, results for FCFS and LCFS are identical and therefore shown in a single column. The exact formulas in Theorem 1 correctly predict the CWS results.

There are two considerations that arise from this test: (a) as shown in the proof of Theorem 1, CWS makes at most 1 error to detect exactly the position of the long jobs (zero errors are committed only if $l = n_s$), which confirms that CWS can react perfectly to the scheduling error by jumping away and without being explicitly instructed to do so, i.e., the decision is taken only on a statistical basis. In comparison, the performance of blind methods degrades remarkably for larger values of n; (b) thanks to the low number of errors, the response times of CWS are approximately the same as SEPT. This suggests near-optimality and provides intuition about the fact that CWS can react effectively to scheduling errors.

Seeking Test

We consider the more challenging seeking test that estimates the capability of CWS of finding regions with short jobs. While the avoidance test is used to show that the algorithm reacts quickly to erroneous scheduling of long jobs, the seeking test shows how quickly the reaction takes to find short jobs. Intuitively, this is a much more difficult test because it is a measure of the cumulative knowledge of the algorithm on the entire workload. To help comparison, we use the same workload model $\mathscr{W}_2$ considered in the avoidance test.

THEOREM 2. *In the seeking test, the number of errors made by* CWS *grows as $O(n/n_s)$.*

Seeking		SEPT		CWS		FCFS/LCFS		RAND	
n	n_l	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$
10	*2*	*0.0*	35	*1.0*	114	*1.0*	114	*1.3*	114
10	*5*	*0.0*	154	*2.0*	238	*2.5*	278	*3.5*	278
10	*8*	*0.0*	362	*3.0*	418	*4.0*	441	*5.2*	442
100	*20*	*0.0*	258	*2.7*	414	*10.0*	1050	*19.7*	1051
100	*50*	*0.0*	1313	*2.9*	1432	*25.0*	2550	*48.9*	2550
100	*80*	*0.0*	3258	*4.6*	3340	*40.0*	4050	*76.2*	4050

Table 3: Seeking test results for different values of the workload length n and of the number of small jobs n_s. The experimental methodology is similar to the avoidance test.

Pattern		SEPT		CWS		FCFS/LCFS		RAND	
n	T	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$	ε	$\mathbb{E}[R]$
10	*2*	*0.0*	154	*0.5*	179	*4.5*	278	*5.0*	278
10	*5*	*0.0*	362	*2.0*	402	*6.0*	441	*5.6*	441
10	*8*	*0.0*	429	*3.5*	465	*5.3*	482	*6.5*	482
100	*20*	*0.0*	4565	*9.5*	4612	*85.5*	4800	*69.6*	4802
100	*50*	*0.0*	4853	*24.5*	4902	*73.5*	4950	*74.8*	4950
100	*80*	*0.0*	4926	*39.5*	4968	*59.3*	4988	*48.4*	4987

Table 4: Pattern test results for different values of the workload length n and of the period T. The experimental methodology is similar to the avoidance test.

PROOF. The proof is reported in the technical report [4]. □

Results for the same examples considered in the avoidance test are given in Table 3. The results shown are again computed by executing the different algorithms on the seeking test workloads: it is easy to verify that the growth in the number of errors in Table 2 follows the theoretical $O(n/n_s)$. Similarly to the avoidance test, the number of errors scales much better in CWS than in the other size-blind policies as the number of jobs increases, yet it is interesting to note that CWS is not very different from the other algorithms if the number of jobs is small ($n = 10$). It seems possible to explain this fact by arguing that if there are few large jobs, then the performance gain can be limited, as confirmed by the fact that for $n_l = 8$ the CWS results are better than for $n_l = 2$. Consistently with this observation, in the harder case $n = 100$, both the number of errors and the response time are very good approximations of SEPT performance and much better than size-blind methods. Note in particular that n/n_s is roughly equal to the mean number of errors that RAND makes before finding the first small jobs, but is also approximately the *total* number of errors of CWS. This illustrates that CWS does not perform a scheduling decision randomly, but instead operates smartly according to information collected from the correlations. From the proof of the approximation in Theorem 2 it also emerges that CWS performs essentially a divide-and-conquer search that at each step divides by two the maximum size of an undetected region of small jobs. This approach is intuitively very effective for early detection of large groups of small jobs.

Pattern Test

The performance of CWS in the pattern test shows a case where the algorithm is extremely efficient in exploiting the dependence structure, but inferring the actual structure of the flow is much harder than exploiting the correlations.

THEOREM 3. *If large jobs are spaced by T small jobs, there exists a non-fully-disjoint workload model $\mathscr{W}_T$ such that* CWS *makes on average $\varepsilon_{\mathsf{CWS}} = (T-1)/2$ errors.*

PROOF. Consider a workload model $\mathscr{W}_T$ with T classes, where the first $T-1$ classes have identical large job distribution $F_l(t)$ and the last class has the small job distribution $F_s(t)$. Define a probability matrix $\boldsymbol{P}$ such that $p_{k,1+mod(k,T)} = 1.0$, then it is trivial to see from Proposition 1 that, after the first small job is found by CWS, then the algorithm knows deterministically from $\boldsymbol{P}$ where the other small jobs are located and hence schedules them first without errors because their expectation is always smaller than the other jobs. However, if the first scheduled job is large, the algorithm has still to determine the currently active state in $\mathscr{W}_T$ when the first job was generated. Noting that any job in position $j > 1$ cannot be more probable of being small than the job in position $j = 2$ due to the identical values of the probabilities in $\boldsymbol{P}$, we conclude easily that CWS works in a FCFS-like fashion until the first small job is found which takes exactly l decisions. Thus the number of errors is always l and averaged on $l = 0, \ldots, n_s$ gives immediately the formulas of $\varepsilon_{\mathsf{CWS}}$. □

The results in Table 4 compare the pattern test performance of the different scheduling policies. The table is given as a function of T which is proportional to the number of large jobs n_l. The number of errors ε grows in the pattern test with a linear trend with respect to the period T. This increased difficulty with respect to the other tests is given by the fact that the workload model $\mathscr{W}_T$ defined in the proof of Theorem 3 has T states which make the estimation of the state from which a job has been sampled harder than in $\mathscr{W}_2$. This is because the model is *not* fully disjoint as $\mathscr{W}_2$ and illustrates the increased difficulty of defining the job size classes as overlapping in terms of job size distribution. However, from the proof of Theorem 3 it still emerges clearly that CWS has no difficulty in dealing with the correlations.

5. SIMULATION RESULTS

We have also studied the performance of CWS in queueing environments using simulation with an estimation window size of length $W = 10^4$. We compare the following policies: CWS, FCFS, LCFS, and SEPT. FCFS and LCFS are expected to be pessimistic bounds on the average performance of CWS. SEPT is instead an optimistic bound on the achievable performance because of its exact a priori knowledge of job class memberships which are instead inferred probabilistically by CWS. The purpose of the experiments in this section is to evaluate whether CWS can approach the performance of a method like SEPT, which relies on exact information about the workload.

We simulate a $M/G/1$ queue using no less than two million samples in each experiment, in heavy load experiments we increase the sample space up to ten million values. For each simulation, the service process is drawn from a K state workload model where the job size distribution of each state is exponential. In order to consider the problem in its full generality, the classes are *not* fully disjoint, i.e., an observed job size t is not uniquely assigned to a single class, thus job size tracking is harder than in Section 4. Throughout the experiments, we vary the squared coefficient-of-variation of service times in $SCV = 3, 9, 20$; mean inter-arrival and service times are scaled in order to examine the system performance under different utilization levels, i.e., $\rho = 0.5$ and $\rho = 0.8$ which represent medium and heavy loads, respectively. We have also tested with $\rho = 0.2$, but in such light load conditions the gap between FCFS, LCFS, and SEPT is often very small which makes it difficult to motivate the need for specialized policies for correlated workloads.

To study the sensitivity of the different scheduling techniques to the workload structure, which in addition to the moments depends

on the number of workload classes K and on the type of correlation considered (e.g., positive or negative decay rates), we consider two illustrative case studies. In the first case study, the workload has $K = 2$ classes and we assume a geometric decay rate of correlations denoted by σ. For $K = 2$ it can be shown that σ is the second largest eigenvalue of $\boldsymbol{P}$. In the experiments, such decay rate is varied in $\sigma = 0, 0.1, \ldots, 0.9$; we do not consider $\sigma = 1$ or close values because these are often degenerate cases where all job sizes in the queue are for extended period of time only large or only small which makes scheduling ineffective. Note also that $\sigma = 0$ corresponds to the case where there are no correlations in the workload, thus CWS is expected to behave as FCFS.

In the second case study, we set $K = 3$ and thus consider three different workload classes. This choice provides a much more expressive family of correlation structures, where the autocorrelation function simultaneously depends on two decay rates σ_1 and σ_2, which are eigenvalues of $\boldsymbol{P}$. This is important because it allows to describe processes with high variability and both positive and negative decay rates which are found in real systems, see for instance the game console READ workload in [18, Fig. 7]. To simplify comparison with the case $K = 2$, we set $\sigma = \sigma_1 = -\sigma_2$, and explore the same range of values $\sigma = 0, 0.1, \ldots, 0.9$ considered in the first case study. Note that mixtures of negative and positive correlations provide a much harder stress case than $K = 2$ which for high correlations is often close to a workload with ON/OFF behavior.

Experimental Results: $K = 2$ Workload Classes

Figure 5 depicts the speedup of CWS, SEPT, and LCFS relatively to FCFS, i.e., the ratio of mean response times under FCFS to the mean response time under the policy being evaluated. We use FCFS as a baseline since in all considered experiments FCFS performance is worse than for the other disciplines. Mean response times are simulated for different utilization, correlation, and variability levels. Our results indicate that increasingly large correlations tend to degrade the system performance under all policies, especially under high-variability and heavy-load utilization. Details about the relative performance of the different methods are given below.

SEPT provides speedups less than 1.6 for $SCV = 3$, however as the variability increases the speedup grows considerably to more than 7.0 for $SCV = 20$ and utilization 0.8. Furthermore, SEPT is much better than the other methods for low correlation values, where CWS and LCFS converge to FCFS average performance (speedup 1.0).

LCFS provides remarkably good improvement over FCFS in many cases. In particular, it is observed that LCFS performance is maximized when the correlation decay rate is 0.9, although small deviations are observed for $SCV = 3$, where nevertheless the three scheduling techniques perform quite closely in the high-correlation case. This seems to suggest that LCFS may be effective for traces with long burst durations, i.e., with ON/OFF behavior, but its performance rapidly degrades in absence of large bursts in the workload. In spite of the good performance of LCFS compared to FCFS, it is found that CWS can be even 200% faster than LCFS under medium/high correlations.

CWS provides very good results under medium and high correlations. In such cases, CWS is extremely close to SEPT under high correlations and it is by far the best technique also under medium correlations, showing that inference can be effective also without the need of strong burstiness. Clearly, low correlations make the inference from observed job sizes progressively more difficult and less informative, thus the reduced performance of CWS in such cases appears unavoidable.

We further investigate in Figure 6 the complementary cumulative distribution function (CCDF) of the response times for different variability levels and for fixed correlation decay rate $\sigma = 0.6$. The graphs show that for most jobs the smallest response time tail is under FCFS and SEPT scheduling. In particular, SEPT provides the best response times for the distribution body and it does not show a significantly different distribution tail compared to FCFS. This effect is consistent across all three experiments in Figure 6. CWS and LCFS have instead longer tails of response times than FCFS and SEPT. The results indicate that LCFS is far more aggressive than CWS in delaying jobs. Note that in this experiment the average response time of LCFS is about 40% slower than for CWS, thus this increased unfairness is not justified by better mean performance.

Experimental Results: $K = 3$ Workload Classes

Figure 7 illustrates experimental results for the workload with $K = 3$ classes and positive and negative decay rates. Similarly to the high-variability cases for $K = 2$, SEPT has a large speedup over FCFS that is again maximized for the largest absolute decay rate value σ. CWS performs slightly less accurately than for $K = 2$, in particular for medium/high values of σ. This is expected due to the increased workload complexity that makes inference of class memberships harder than with 2 classes. However, the results indicate that CWS provides strong performance across a wide range of correlations and it is again the method of choice in absence of exact a priori information. The performance of LCFS appears degraded for $SCV = 20$ compared to the case $K = 2$ and indicates that LCFS performance may decrease as the structure of the correlations becomes more complex.

6. RELATED WORK

CWS prioritizes jobs that are expected to be short based on information obtained from past scheduling history and on the HMM workload model $\mathscr{W}$. Policies which possess features similar to CWS have been investigated in recent work, both within the scope of independent and correlated workloads, this section gives a comparative analysis of these methods with respect to CWS.

Policies for independent workloads. Scheduling policies that favor short jobs, such as SRPT, have been found not to penalize large jobs too much [22] and are very effective because large jobs significantly degrade the tail of response times in queueing environments [2]. Starting from these observations, new scheduling approaches which prioritize short jobs based on exact job size information have been recently investigated, such as the fair share protocol (FSP) [9]. However, because exact information can be hard to obtain in certain settings, current research is trying to define new classes of policies that do not need exact values. These include the recently-defined ϵ-SMART policies [23], the class-based SRPT approximation in [12], or SEPT and WSEPT which schedule jobs based on size class expectations [15, 20]. Compared to these policies, CWS shares the goal of favoring short jobs based on partial information about job sizes. Because of the use of expectations, CWS may be seen as a broad generalization of SEPT where conditional job size distributions are inferred. A limitation is that CWS is non-preemptive, thus it is not possible to consider remaining processing times.

Policies for correlated workloads. To the best of our knowledge, very few general scheduling results exist for correlated workloads. Some authors have explicitly attributed this lack to the extreme difficulty in developing effective policies for models with depen-

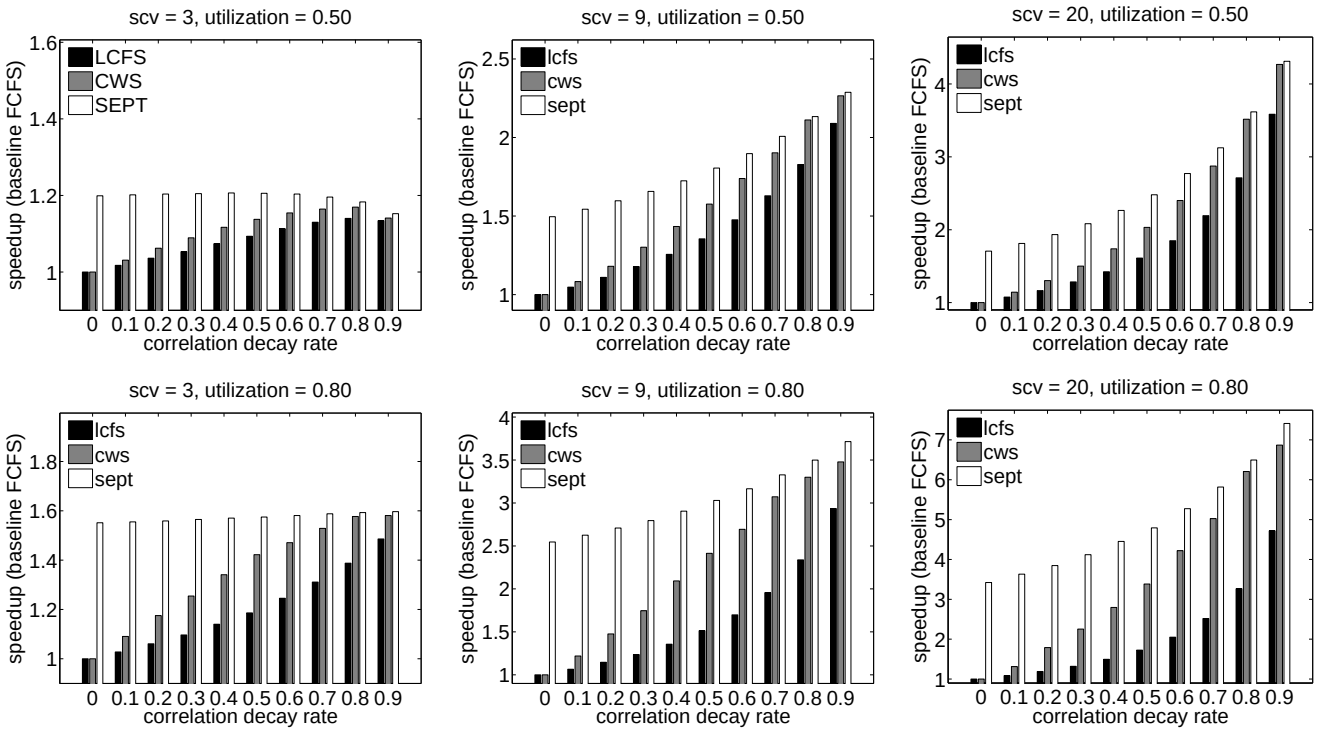

Figure 5: $K = 2$ workload classes: comparison of CWS, LCFS, and SEPT speedup over FCFS for different utilizations, autocorrelation decay rates, and variability levels. The estimation window size is $W = 10^4$.

dence [3]. In [3], the authors also prove sub-optimality bounds for normally distributed job sizes under equicorrelation conditions. Dependence among jobs in scheduling has been mostly explored in a deterministic setting focusing on graph-based precedence constraints [5]. From a practical perspective, however, correlations are accounted in several applications. The approach in [21] defines a scheduling strategy for broadcast systems which uses first-order correlations between data items. In [13], a frame-level model of variable bit rate (VBR) streams is obtained which describes correlations by the recursive TES method [11]. In [25] an online real-time energy-aware scheduling model is presented which can account for job covariances and is solved by optimization methods. Compared to these works, CWS can be cheaper computationally, but most importantly it is application-independent.

7. CONCLUSION

We have presented CWS, a scheduling technique that favors short jobs based on job size predictions obtained from a HMM workload model and past scheduling history. We have shown by simulation that CWS can often approximate effectively SEPT, which assumes exact a priori knowledge about the workload. Further, in presence of correlated workloads CWS performance is significantly better than for policies such as FCFS or LCFS. Simulation results show that there exist several correlation levels where the CWS algorithm is very effective and occasionally nearly optimal. Such results are also confirmed on fixed-size workloads where we have characterized CWS performance using analytical expressions.

Several extensions of this work are possible. A generalization of CWS to the preemptive case appears possible, although the impact on computational costs should be assessed. Further work is also needed to establish if metrics other than size expectation exist that can help in scheduling correlated workloads. Finally, it would be useful to develop techniques for online adaptation of the HMM workload model.

Acknowledgement

This work was supported by NSF grants CCF-0811417 and CCF-0937925 and by the Imperial College Junior Research Fellowship. The authors thank the anonymous referees for helpful suggestions that improved the experimental part of this paper.

8. REFERENCES

[1] A. T. Andersen and B. F. Nielsen. A Markovian approach for modeling packet traffic with long-range dependence. *IEEE JSAC*, 16(5):719–732, 1998.

[2] S. C. Borst, O. J. Boxma, R. Núñez Queija, and A. P. Zwart. The impact of the service discipline on delay asymptotics. *Performance Evaluation*, 54(2):175–206, 2003.

[3] R.J. Boys, K.D. Glazebrook, and C.M. McCrone. Single machine scheduling when processing times are correlated normal random variables. *European Journal of Operational Research*, 102:111–123, 1997.

[4] G. Casale, N. Mi, E. Smirni. CWS: a Model-Drivel Correlated Workload Scheduler. Technical Report WM-CS-2010-05, College of William and Mary, Department of Computer Science, 2010.

[5] J. R. Correa and A. S. Schulz. Single machine scheduling with precedence constraints. Technical Report MIT Sloan Working Paper No. 4499-04, Massachusetts Institute of Technology (MIT) - Sloan School of Management, August 2004.

[6] A. Erramilli. Performance impacts of self-similarity in traffic. In *Proc. of joint ACM SIGMETRICS/IFIP Performance*, pages 265–266, New York, NY, USA, 1995. ACM.

[7] D. G. Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. Online preprint, Jun 2008.

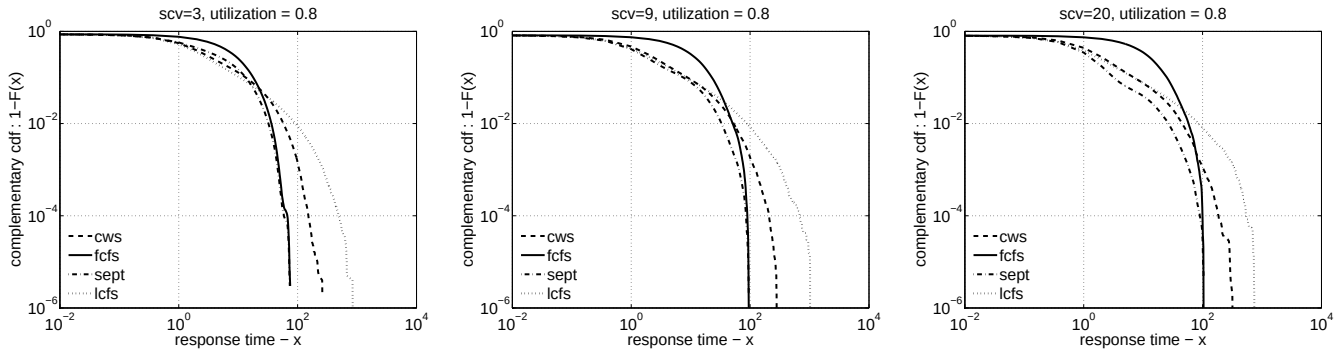

Figure 6: CCDF of response times for a network with $\sigma = 0.6$ and same utilization and variability levels in Figure 5. The estimation window size is $W = 10^4$.

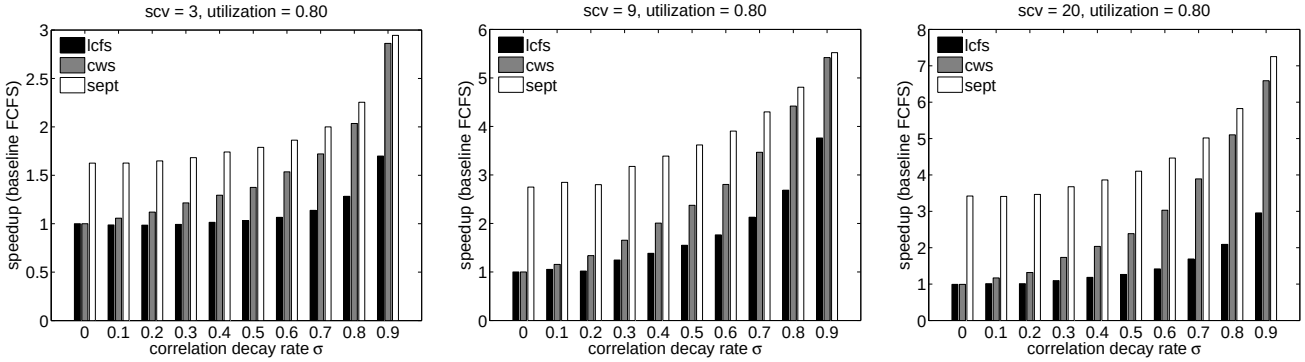

Figure 7: $K = 3$ workload classes: comparison of CWS, LCFS, and SEPT speedups over FCFS, window size $W = 10^4$.

[8] H. Feng, V. Misra, and D. Rubenstein. PBS: a unified priority-based scheduler. In *Proc. of ACM SIGMETRICS*, pages 203–214, 2007.

[9] E. J. Friedman and S. G. Henderson. Fairness and efficiency in web server protocols. In *Proc. of ACM SIGMETRICS*, pages 229–237, 2003.

[10] A. Heindl. *Traffic-Based Decomposition of General Queueing Networks with Correlated Input Processes*. Ph.D. Thesis, Shaker Verlag, Aachen, 2001.

[11] D. L. Jagerman and B. Melamed. The transition and autocorrelations structure of TES processes. *Stochastic Models*, 8:193–219, 1992.

[12] P. R. Jelenkovic, X. Kang, and J. Tan. Adaptive and scalable comparison scheduling. In *Proc. of ACM SIGMETRICS*, pages 215–226, 2007.

[13] A. A. Lazar, G. Pacifici, and D. E. Pendarakis. Modeling video sources for real-time scheduling. *Multimedia Systems*, 1:835–839, 1993.

[14] W. S. Lovejoy. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175, 1991.

[15] R. H. Möhring, A. S. Schulz, and M. Uetz. Approximation in stochastic scheduling: the power of lp-based priority policies. *Journal of the ACM*, 46(6):924–942, 1999.

[16] R. D. Nelson. The mathematics of product form queuing networks. *ACM Computing Surveys*, 25(3):339–369, 1993.

[17] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[18] A. Riska, E. Riedel. Long-Range Dependence at the Disk Drive Level. *Proc. of QEST*, 41–50, 2006.

[19] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, 1992.

[20] M. Scharbrodt, T. Schickinger, and A. Steger. A new average case analysis for completion time scheduling. *Journal of the ACM*, 53(1):121–146, 2006.

[21] W. Uchida, T. Hara, and S. Nishio. Scheduling correlated broadcast data considering access frequencies with temporal variations. In *Proc. of IEEE NCA*, page 89, 2003.

[22] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proc. of ACM SIGMETRICS*, pages 238–249. ACM, 2003.

[23] A. Wierman and M. Nuyens. Scheduling despite inexact job-size information. In *Proc. of ACM SIGMETRICS*, pages 25–36, New York, NY, USA, 2008. ACM.

[24] X. Yuan and M. Ilyas. Modeling of traffic sources in atm networks. In *Proc. of SoutheastCon*, pages 82–87, 2002.

[25] X. Zhong and C. Z. Xu. Energy-aware modeling and scheduling of real-time tasks for dynamic voltage scaling. In *Proc. of IEEE RTSS*, pages 366–375, 2005.