p-YDS Algorithm: An Optimal Extension of YDS Algorithm to Minimize Expected Energy For Real-Time Jobs*

TIK Report No. 353

Pratyush Kumar and Lothar Thiele
Computer Engineering and Networks Laboratory, ETH Zurich

July 31, 2014

Abstract

The YDS algorithm computes a schedule on a DVS-enabled resource to meet deadlines of all jobs and optimally minimize the total energy consumption. The algorithm requires that an exact execution time of each job be known. For settings where execution times are variable or uncertain, *stochastic scheduling* has been proposed to preferentially accelerate less probable phases of jobs to reduce the *expected* energy consumption. However, an analogue to the YDS algorithm for the stochastic setting has not been optimally solved. In this paper, we propose the p-YDS algorithm to minimize the expected energy consumption for a set of jobs with arbitrary arrival times, deadlines, and execution times. We then derive the competitive ratio of the YDS algorithm w.r.t. the p-YDS algorithm, for the metric of expected energy consumption. By comparing two optimal algorithms, this ratio specifies the worst-case energy cost of being agnostic to the variability in the execution time of jobs.

1 Introduction

Managing energy consumption of an electronic device is a primary design challenge. This applies to a variety of devices, from battery-operated mobile and remote devices, to powerful servers deployed in data centers. A natural counterpoint to energy minimization is the guaranteeing of performance of the supported applications. Indeed, higher performance may only be possible at an expense of higher energy consumption. A mainstream research challenge, over the last several decades, has been to explore this trade-off. Characteristically,

*This is an extended version of our EMSOFT 14 paper by the same title with DOI: `http://dx.doi.org/10.1145/2656045.2656065`.

this research has been cross-layer and has focused on various aspects such as circuit design, architecture design, operating systems, and battery technology.

In this work, we study scheduling of different jobs by the operating system as a technique to minimize energy consumption while (a) meeting real-time deadlines of timing-critical software, and (b) taking advantage of the, now standard, feature of dynamic voltage scaling (DVS) on a uniprocessor system. On DVS-capable processors, the operating voltage and the processing speed of a processor can be increased or decreased. Increasing the speed results in improved processing capability, but at higher power consumption. Furthermore, in typical CMOS-based devices, the energy spent to provide a unit of work (such as a processor cycle) is higher at a higher speed, i.e., power consumption is a convex increasing function of the speed. In such DVS-capable processors, the speed and the scheduling of jobs can be carefully co-designed to minimize the energy consumption while satisfying timing guarantees. We refer to this co-design as *speed-scheduling*.

Different formulations of speed-scheduling for real-time systems have been well studied and some clear principles have been identified. For a single job with a known execution time, if the speed is chosen such that the job finishes exactly at its deadline, then the energy consumed is minimized. If such a speed is not available, then the two closest speeds need to be duty-cycled. Similarly, if multiple periodic *tasks* with respective deadlines equal to periods are scheduled under the Earliest Deadline First (EDF) policy, then the speed must be chosen such that the *effective utilization* is one. In these cases, if the relationship between speed and power consumption is known, then closed form solutions identify the optimal speed-schedule.

However, the problem does not admit a direct solution when either the tasks are not periodic, or have relative deadlines unequal to their periods. For such problem instances, Yao, Demers, and Shenker [1] identified an algorithm to compute the speed schedule, which has since come to be known as the YDS algorithm. Several years later, the YDS algorithm was shown to optimally minimize the energy consumption [2]. The YDS algorithm iteratively computes the speeds to be used in different parts of the schedule by defining a notion of *intensity* of intervals. The ingenious algorithm illustrates that inter-task optimization of a speed-schedule is not trivial.

In all the above cases, a *constant* execution time of each job is assumed to be known. However, in reality two problems arise: execution time of a job is (a) not constant, and (b) not known exactly. One response to this *variability* and *uncertainty*, is to characterize each job with a worst-case execution time (WCET). This choice guarantees that the real-time constraints are met by the speed-schedule for all observable execution times. Furthermore, this choice also minimizes the *worst-case energy consumption*. In certain devices, the worst-case energy consumption could be a significant metric: For instance, to ensure that a battery-operated device does not run out of power between regular charges. However, in many other devices, it may be more important to minimize the *average* or *expected* energy consumption.

To be able to compute the expected energy consumption, and design a speed-

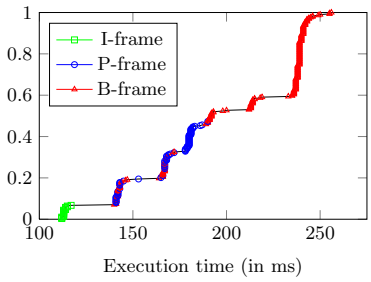

Figure 1: CDF of the execution time in processing a frame in H.264 encoder. Different frame-types contribute to variation in the execution time.

schedule to optimize it, statistics of the execution times of jobs are required. A primary statistic of the execution time is the *cumulative distribution function* denoted as CDF. Specifically, $\texttt{CDF}(c)$ denotes the probability that the execution time exceeds c. Such profiling was used to characterize jobs from different application domains such as multimedia [3], computer vision [4], and word processing [5].

To consider a concrete example, we profiled the H.264 encoder sourced from the MediaBench II benchmark suite [6]. 300 frames of a video were decoded and the CDF of the frame-wise execution time is as shown in Figure 1. The H.264 standard includes different frame-types, encoding which takes markedly different amounts of time. Furthermore, frames of the same type exhibit a range of data-dependent execution times. In effect, this results in the execution time varying by up to 2.5×. From the CDF, we can compute an approximate discrete *probability density* by partitioning the execution of a job into *phases*. A phase is a sequential portion of a job which begins and ends at fixed execution times, denoted as $c_{\min}$ and $c_{\max}$, respectively. In addition, a phase has a probability of execution, given by $1 - \texttt{CDF}(c_{\min})$, i.e., the probability of a phase is the probability that the execution time exceeds $c_{\min}$. For the H.264 encoder, consider dividing the execution of the encoder into 6 phases as shown in the x-axis of Figure 2. The histogram represents the probability of the chosen phases.

With such statistics, we can hope to reduce the expected energy. In particular, less probable phases can be run faster while more probable phases are run slower. This potential has been clearly observed in several works. The earliest reference, to our knowledge, emerges in [7], where the speed is dynamically changed based on the *average* execution time of the remaining jobs. A practical demonstration of such scheduling for realistic applications was presented in [5]. For a single job with known probability distribution, and given hard real-time deadline, the schedule that optimally minimizes the average energy consumption was presented in [8] and [3]. Generally, these results were referred to as *intra-task stochastic scheduling*. The first extension to multiple tasks was presented in [9] where a frame-based task model was considered. Task-sets with multiple periodic tasks with deadlines equal to periods scheduled under

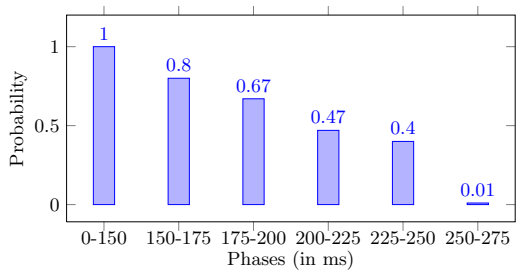

Figure 2: Approximate discretized probability density for the H.264 encoder application.

EDF policy, were studied in [4]. The authors show that an analytical formula generates the optimal static speed-schedule for an implicit deadline periodic task-set. Detailed study of realistic effects such as overhead in switching speeds and leakage-aware power models have been considered in [10] and [11]. An additional dimension is the dynamic slack distribution amongst tasks to consider inter-task effects [7, 12, 13].

The existing work does not solve the following problem formulation. Neglect overheads and leakage power, and restrict to only static scheduling. For a given set of jobs with arbitrary arrival times, deadlines and `CDF`, compute the static speed-schedule to (a) meet the deadlines of all jobs in the worst-case, and (b) minimize the expected energy. Existing work on stochastic scheduling has only considered periodic tasks with implicit deadlines. As evidenced from the non-stochastic case with the YDS algorithm, considering jobs with non-periodic arrivals or periodic tasks with deadline unequal to periods is not straight forward. We aim to extend stochastic scheduling by designing an algorithm analogous to the YDS algorithm. The core problem here is to derive how tasks individually speed-scheduled according to their `CDF` share the available time with each other.

In this paper, we make the following contributions.

1. We propose p-YDS algorithm to solve the said problem. To this end, we define a notion of *interference* between jobs and identify when a job can interfere with another job. Motivated by this definition, we design p-YDS algorithm and prove its optimality.
2. We study the competitive ratio of the YDS algorithm in comparison to the p-YDS algorithm to quantify the worst-case energy cost of being agnostic to variability and uncertainty in the execution time. To this end, we derive an analytical bound on the competitive ratio. Significantly, this ratio does not depend on the job parameters like arrival times, WCETs, and deadlines.
3. Higher speeds required by p-YDS algorithm may not always be available. To counter this, we optimally extend the algorithm for a given maximum speed.

4. We propose a method to compare the algorithms for a *given* job-set. We show that we can compute an *energy ratio* for each application. For any job-set the energy cost of YDS relative to p-YDS is upper-bounded by the maximum of the energy ratios of the constituent jobs, independent of other parameters.

The rest of the paper is organized as follows. In Section 2, we define the system model. We propose the p-YDS algorithm and prove its optimality in Section 3. We compute the competitive ratio of the YDS algorithm in Section 4. We present extensions to the p-YDS algorithm in Section 5. In Section 6 we present experimental results, and conclude in Section 7.

2 System Model

Processor Model We consider a single-core processor whose speed can be varied continuously. In Section 5, we extend the work to consider a given upper-bound on the speeds. Power consumption includes both static and dynamic components. We restrict our attention to dynamic power, which is proportional to s^α, at speed s, for some $\alpha \geq 2$. We do not consider any time or energy overhead in switching between the speeds.

Job Model The job-set contains N independent fully preemptible jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_N\}$. As discussed in the previous section, the execution times of jobs are variable or uncertain. To characterize this, each job, say J_i, is divided into phases and is characterized by the following two sets of parameters.

1. The *standard parameters* are $(a_i, d_i, c_i^{\max})$, where,
   - a_i is the arrival time,
   - d_i is its deadline, and
   - $c_i^{\max}$ is the WCET.
2. The *stochastic parameters* are $(n_i, \mathbf{c_i}, \mathbf{p_i})$, where
   - n_i is the number of phases,
   - $\mathbf{c}_i$ is a vector of n_i numbers, with c_{ij} as the fixed execution time of the jth phase satisfying the WCET, i.e., $c_i^{\max} = \sum_{l=1}^{n_i} c_{il}$, and
   - $\mathbf{p}_i$ is a vector of n_i numbers, with p_{ij} as the probability of execution of the jth phase, i.e., the probability that the execution time exceeds $\sum_{l=1}^{j-1} c_{il}$.

Execution times are specified for a fixed normalized speed and scale linearly with speed. Specifically, the time to execute a job with WCET c at speed s is (c/s).

The described job model allows us large flexibility in modeling of real-time job-sets. We can model periodic tasks by the explicit jobs within the hyper-period. We can model bursty or sporadic tasks, or even non-recurring jobs. Additionally, for periodic tasks, the deadlines can be smaller, equal, or larger than the periods.

Static Speed-Schedule A static speed-schedule identifies for each time-instance (a) a certain job to be executed, and (b) the speed of execution. The speed-schedule is static in the following sense: During execution, if a certain phase of a job does not execute, then the processor idles and consumes zero (dynamic) power, i.e., there is no dynamic slack reclamation. We say a static speed-schedule is *feasible* if all jobs meet their deadlines, independent of which phases execute.

Energy Metric For a given static speed-schedule and a job-set, depending on which phases of the jobs execute, the total energy consumption would vary. Let the energy consumption as a function of the execution times of the N jobs, $c_1, c_2, \ldots, c_N$, be given as $E(c_1, c_2, \ldots, c_N)$. Let $\texttt{CDF}'_i$ denote the derivative of the cumulative frequency distribution of job $J_i \in \mathcal{J}$. Then the expected energy $\overline{E}$ is given as follows.

$$\int_{c_1} \ldots \int_{c_N} E(c_1, \ldots, c_N) \texttt{CDF}'_1(c_1) \ldots \texttt{CDF}'_N(c_N) \mathrm{d}c_N \ldots \mathrm{d}c_1$$

For a static speed-schedule and the execution of jobs discretized into phases, the above integrals becomes finite sums. Let $E_p(ij)$ denote the energy consumed in executing the jth phase of task J_i. Then the $\overline{E}$ is given as follows.

$$\overline{E} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} p_{ij} \times E_p(ij) \tag{1}$$

The above formulation is significant in the following sense: With regards to the metric $\overline{E}$, the execution of a phase of a job can be treated as an independent random variable with a known probability. This stochastic property of the metric $\overline{E}$ must be contrasted from the fact that a phase can execute only if all previous phases execute.

Problem Definition Given a power model and the job-set $\mathcal{J}$, the problem is to identify a static speed-schedule such that

- all jobs meet their deadlines independent of which phases execute, and
- the expected energy consumption $\overline{E}$ is minimized.

The above problem definition reflects the nature of the problem. On the one hand, timing guarantees are to be given in the worst-case. On the other hand, when optimizing energy we focus on the average-case.

3 The p-YDS Algorithm

In this section, we propose p-YDS to optimally solve the problem of minimizing expected energy consumption. First, we present a set of necessary conditions that an optimal static speed-schedule must satisfy. Then, using these conditions, we design the optimal algorithm. Finally, we illustrate it with a running example.

3.1 Necessary Conditions of an Optimal Static Speed-Schedule

A static speed-schedule must identify (a) the exact time intervals within which the jobs execute, and, (b) the speeds with which different parts of the jobs execute. We will now derive necessary conditions on these two sets of parameters.

Given the speeds of executing the jobs, the scheduling policy does not affect the value of $\overline{E}$. Consequently, we can use the optimal scheduling policy for meeting deadlines of jobs, i.e., the Earliest Deadline First (EDF) policy. This leads to the following condition.

Lemma 1 *In an optimal static speed-schedule, jobs are scheduled according to the Earliest Deadline First (EDF) scheduling policy.*

Proofs of some results are in the appendix.

With the scheduling policy fixed, we are hence concerned only with the speeds. The next question relates to how often the speed must be changed when executing a job. From the convexity of the power function, a constant speed optimally minimizes the energy consumption when executing a fixed number of cycles. This leads to the following condition.

Lemma 2 *In an optimal static speed-schedule, the speed of execution throughout a phase of any job is constant.*

From the above result it follows that the optimal speed-schedule is specified by the speed of each phase of every job. Hence, we define a specific notation.

Definition 1 (Speed of a Phase) *In an optimal speed-schedule, let the speed of execution of the jth phase of job J_i be denoted as s_{ij}.*

The next question relates the speeds of different phases of a job. As has been similarly observed in [3,8], for a particular job the speeds of different phases are related.

Lemma 3 *In an optimal static speed-schedule, the speeds of execution of any two phases of any job J_i satisfy*

$$\frac{s_{ij}}{s_{ik}} = \sqrt[\alpha]{\frac{p_{ik}}{p_{ij}}}, \quad \forall\ j,k \in \{1,\ldots,n_i\}. \tag{2}$$

From the above result, a phase with a lower probability of execution is to be executed at a higher speed. In other words, the later phases of jobs, which are less likely to be executed, are accelerated to complete sooner. Thus, it follows that the optimal speed-schedule is specified by a *single* speed for each job. Hence, we define a specific notation.

Definition 2 (nominal-speed of a Job) *In an optimal static speed-schedule, the nominal speed $\overline{s_i}$ of a job J_i satisfies the following for all phases j of that job*

$$\overline{s_i} = s_{ij} \times \sqrt[\alpha]{p_{ij}}. \tag{3}$$

To conclude this part, we observe that an optimal speed-schedule is fully specified by the nominal-speed for each job. Given the nominal-speed, we can compute the constant speeds for the individual phases, and execute jobs according to the EDF scheduling policy.

3.2 The Eligibility to Interfere

The above results relate to the conditions on the optimal speed-schedule based on properties of a single job. As observed before, the most critical part is in understanding how multiple jobs with arbitrary arrival times and deadlines share the available time on a shared resource. To study this carefully, we formalize a notion of interference.

Definition 3 (Interference) *In a given speed-schedule, we say that job J_i interferes with job J_j, if and only if, J_i executes within the arrival time and deadline of J_j, i.e., within the interval $[a_j, d_j]$.*

An interfering job reduces the amount of time available to execute the interfered job. Hence, the next question is to identify conditions under which a job can interfere another job in the optimal speed-schedule. We answer this in the following crucial result.

Theorem 1 (Interference Condition) *For the optimal static speed-schedule, if J_i interferes with J_j then the nominal-speeds of the two jobs are related as $\overline{s_i} \geq \overline{s_j}$.*

Proof: We prove this by contradiction. For the two defined jobs J_i and J_j such that J_i interferes with J_j, let $\overline{s_i} < \overline{s_j}$ for some speed-schedule $\mathcal{S}$. Let E_i, E_j denote the expected energy consumption for jobs J_i and J_j, respectively. E_i (and similarly E_j) is given as

$$E_i = \sum_{k=\{1,\ldots,n_i\}} p_{ik} \cdot c_{ik} \cdot \left(\frac{\overline{s_i}}{\sqrt[\alpha]{p_{ik}}} \right)^{(\alpha-1)}. \tag{4}$$

Consider modifying the schedule $\mathcal{S}$, without changing the schedule for any job other than J_i and J_j. Then, the total time for which J_i and J_j execute must be a constant, i.e.,

$$\sum_{k=\{1,\ldots,n_i\}} \frac{c_{ik}}{\overline{s_i}/\sqrt[\alpha]{p_{ik}}} + \sum_{k=\{1,\ldots,n_j\}} \frac{c_{jk}}{\overline{s_j}/\sqrt[\alpha]{p_{jk}}} = \text{constant}. \tag{5}$$

The derivative of $E_i + E_j$ w.r.t. $\overline{s_i}$ while satisfying (5) is

$$\frac{d(E_i + E_j)}{d\overline{s_i}} = \eta \cdot (\overline{s_i}^{(\alpha-2)} - \overline{s_j}^{(\alpha-2)}),$$

where η is a positive constant. For $\alpha \geq 2$ and $\overline{s_i} < \overline{s_j}$, the above quantity is negative. Thus, we can reduce the total energy by increasing $\overline{s_i}$ and correspondingly decreasing $\overline{s_j}$ to ensure (5). This can be done until either (a) $\overline{s_i} = \overline{s_j}$,

or (b) J_i does not execute within $[a_j, d_j]$ due to the increase in $\overline{s_i}$, whichever condition comes first. Neither case contradicts the interference condition.

We iteratively apply this process for any pair of jobs, and improve $\mathcal{S}$. Ultimately, no pair of jobs would violate the interference condition, if an optimal schedule exists. □

Two points follow from the interference condition.

- For two jobs J_i and J_j, if $a_i, d_i \leq a_j$, then the jobs cannot interfere with each other. Then the condition does not relate the nominal-speeds of such jobs.
- For two jobs J_i and J_j, if $a_j \leq a_i < d_i \leq d_j$, then J_i must interfere with J_j on any feasible speed-schedule. Then the condition enforces that $\overline{s_i} \geq \overline{s_j}$.

For pairs of jobs not belonging to the above extreme cases, the interference condition restricts the class of optimal schedules. We utilize this to design the p-YDS algorithm.

3.3 Algorithm and Optimality

So far, we have established different necessary conditions on the optimal speed-schedule. What remains to be done is to identify a speed-schedule satisfying all the necessary conditions. We will show that satisfying all the necessary conditions defines a sufficient condition for optimality.

The intuition behind the algorithm is the interference condition of Theorem 1 satisfied by the optimal schedule. A job with a highest nominal-speed cannot be interfered by any other job. This motivates a strategy whereby we first compute the speed-schedule for this job. Then, we "remove" this job from the input formulation. Then we iteratively compute the speed-schedule for the job with the next highest nominal-speed, and so on. Indeed, this strategy works and this is illustrated in the pseudo-code in Algorithm 1.

The algorithm computes the nominal-speeds of *intervals* assuming that all jobs arriving and having deadlines within this interval (a) have the same nominal-speed, and (b) execute throughout the interval. The interval with the highest such speed is assigned the speed, and we iterate through the remaining intervals. This results in the following property of the algorithm.

Lemma 4 *In the p-YDS algorithm the nominal-speed assigned to jobs in every iteration is non-increasing.*

Based on this property, we derive the result on the optimality of the algorithm.

Theorem 2 *For any given job-set, the static speed-schedule computed by the p-YDS algorithm minimizes the expected energy consumption while meeting all deadlines.*

Proof: We prove this in two steps. First, we show that the computed speed-schedule guarantees the necessary conditions. Second, we show that modifying

Algorithm 1: p-YDS algorithm

Input: The set of jobs $\mathcal{J}$ and their parameters
Output: The nominal-speed $\overline{s_i}$ for each job $J_i \in \mathcal{J}$ to minimize the expected energy

foreach $J_i \in \mathcal{J}$ **do**
  $\overline{c_i} \leftarrow \sum_{j=1,\ldots,n_i} c_{ij} \sqrt[\alpha]{p_{ij}}$
while $\mathcal{J} \neq \phi$ **do**
  $\max_s \leftarrow 0$
  foreach $J_j \in \mathcal{J}$ **do**
    foreach $J_i \in \mathcal{J} \wedge a_i \geq a_j \wedge d_i \geq d_j$ **do**
      $I_{\mathtt{beg}} \leftarrow a_j$
      $I_{\mathtt{end}} \leftarrow d_i$
      $I_c \leftarrow 0$
      $I_J \leftarrow \phi$
      foreach $J_k \in \mathcal{J}$ **do**
        if $a_k \geq I_{\mathtt{beg}} \wedge d_k \leq I_{\mathtt{end}}$ **then**
          $I_J \leftarrow I_J \cup \{J_k\}$
          $I_c \leftarrow I_c + \overline{c_k}$
      $s_I \leftarrow I_c/(I_{\mathtt{end}} - I_{\mathtt{beg}})$
      if $s_I \geq \max_s$ **then**
        $\max_s \leftarrow s_I$
        $\max_J \leftarrow I_J$
  foreach $J_i \in \max_J$ **do**
    $\overline{s_i} \leftarrow \max_s$
  $\mathcal{J} \leftarrow \mathcal{J} \setminus \max_J$
  foreach $J_i \in \mathcal{J}$ **do**
    $a_i \leftarrow (a_i - \min(\max(0, a_i - I_{\mathtt{beg}}), I_{\mathtt{end}} - I_{\mathtt{beg}}))$
    $d_i \leftarrow (d_i - \min(\max(0, d_i - I_{\mathtt{beg}}), I_{\mathtt{end}} - I_{\mathtt{beg}}))$

the speed-schedule while maintaining the necessary condition cannot reduce the expected energy.

Necessary condition. Note that we implicitly conform to the necessary conditions of Lemmas 1 to 3: We only compute the nominal-speed of each phase in the algorithm. The interference condition of Theorem 1 is satisfied because (a) the nominal-speeds assigned with increasing iterations are monotonically non-increasing (Lemma 4), and (b) jobs assigned speeds in later iterations do not interfere with jobs assigned speeds in earlier iteration.

Optimality. Now consider some other speed-schedule S' which satisfies the necessary conditions of Lemmas 1 to 3 and Theorem 1, and deviates from the schedule S computed by p-YDS algorithm. Let $\overline{s_i}$ and $\overline{s_i}'$ denote the nominal-speeds of any job J_i in schedules S and S', respectively. Let J_k be the job with the highest $\overline{s_k}$, such that $\overline{s_k} \neq \overline{s_k}'$.
Case (a): $\overline{s_k}' < \overline{s_k}$. Given that S satisfies Lemma 4, J_k experiences interference only from jobs J_i with $\overline{s_i} \geq \overline{s_k}$. All such jobs have the same speed in S'. Thus, with $\overline{s_k}' < \overline{s_k}$, J_k will not complete within its deadline in S'.
Case (b): $\overline{s_k} < \overline{s_k}'$. Let the generated slack (w.r.t. S) in S' be reclaimed by job J_l. Since, J_l interferes with J_k, from Theorem 1, $\overline{s_l}' \geq \overline{s_k}'$. Also, from the definition of k, we have $\overline{s_l} < \overline{s_k}$, and therefore $\overline{s_l}' > \overline{s_l}$. Thus, from S to S', increasing the nominal-speed of one job (J_k) necessitates that the nominal-speed of at least one other job be increased. Applying this recursively, we arrive at a

	Standard Params.			Stochastic Params.			
J_i	a_i	d_i	$c_i^{\max}$	n_i	(c_{i1}, p_{i1})	(c_{i2}, p_{i2})	(c_{i3}, p_{i3})
J_1	0	8	6	2	(3, 1)	(3, 1/27)	-
J_2	5	16	7	3	(1, 1)	(2, 1/8)	(4, 1/64)
J_3	15	25	9	3	(1, 1)	(2, 1/8)	(6, 1/27)

Table 1: Standard and stochastic parameters of jobs for Example 1.

contradiction. □

Illustrating Example

Example 1 *Consider the job-set shown in Table 1. It has three jobs, arriving at different times. J_1 has 2 phases, while J_2 and J_3 have 3 phases. Let $\alpha = 3$, i.e., let $P(s) \propto s^3$.*

p-YDS algorithm. The speed-schedule computed by p-YDS algorithm is shown in Figure 3(a). In the first iteration, the nominal-speed is maximum for the interval $[0, 8]$. The only job in this interval, J_1, is executed to cover the entire interval with $\overline{s_1} = 0.5$. Within the job, the speed of the less probable phase is higher. In the next iteration, the nominal-speed is maximum for the interval $[8, 25]$ which includes jobs J_2 and J_3. Both jobs are assigned the nominal-speed 0.412. The expected energy consumption is $\overline{E}_{\text{p-YDS}} = 2.19$.

Now we illustrate the satisfaction of the interference condition (Theorem 1). Only two pairs of jobs can interfere: J_1 and J_2, and J_2 and J_3. According to the schedule we have $\overline{s_1} > \overline{s_2} = \overline{s_3}$. Thus, J_2 cannot interfere with J_1, while no condition is implied on the pair J_2 and J_3. This is indeed true: J_1 executes in $[0, 8]$ and does not experience any interference from J_2.

YDS algorithm. Now consider the speed-schedule computed by the YDS algorithm, which does not distinguish between different phases of a job. The schedule is shown in Figure 3(b). Interval $[15, 25]$ has the highest *density*, and job J_3 is assigned the speed 1, which is the same for all phases. The next chosen interval is $[0, 15]$ where both jobs J_1 and J_2 are assigned the same speed of 0.867. Note the contrast to the speed-schedule computed by the p-YDS algorithm. In particular, the two algorithms differ in which jobs interfere with which other jobs. The expected energy consumption is $\overline{E}_{\text{YDS}} = 4.52$. The ratio of the expected energy consumption for the two algorithms is $\omega = \overline{E}_{\text{YDS}}/\overline{E}_{\text{p-YDS}} = 2.06$.

4 Competitive Ratio of the YDS algorithm

The YDS algorithm computes the optimal speed-schedule to minimize energy consumption while meeting deadlines of all jobs for fixed execution times. For jobs with phases of execution, the algorithm does not minimize the expected

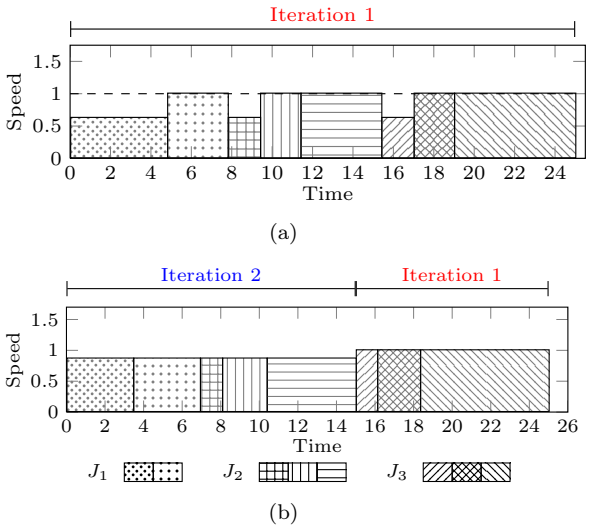

Figure 3: Speed-schedules for the job-set of Example 1 computed by (a) p-YDS and (b) YDS algorithms. Apart from the speeds, the two algorithms differ in assigning higher priority to different parts of the schedule, as evidenced by the intervals in the different iterations.

energy. However, reducing probability of execution of the phases also decreases the expected energy consumed by the speed-schedule computed by the YDS algorithm. A natural question is how far from optimal is the YDS algorithm. We can quantify this, against the optimal p-YDS algorithm. To this end, we define and study a competitive ratio.

4.1 Definition

In general, competitive ratio Ω is defined as

$$\Omega(y) = \max_{x \in X} \frac{\text{Perf. of algo.1 which knows only } y}{\text{Perf. of algo.2 which knows } x \text{ and } y}, \tag{6}$$

where x is a set of problem parameters preferentially known to one algorithm, and X is the set of all possible values of x. Such a competitive ratio helps identify the *worst-case* cost of not knowing the problem parameters x. As a matter of notation, the quantity inside the max operator in (6) is denoted as $\omega(x, y)$. Thus, ω denotes the relative performance of the algorithms for a given problem instance, while Ω denotes the competitive ratio.

In our setting of comparison of the two algorithms, the YDS algorithm knows only the standard parameters of the job-set, while the p-YDS algorithm knows both the standard and stochastic parameters. We would like to compute the competitive ratio which illustrates the worst-case cost of not knowing the stochastic parameters. To this end, we interpret the quantities in (6) as follows.

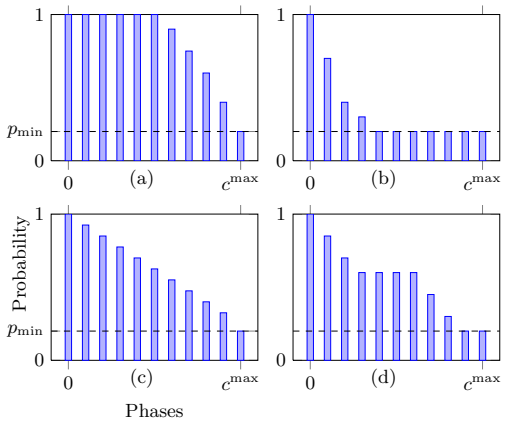

Figure 4: Different sets of stochastic parameters under given constraints. (a) Gaussian-like distribution with mean closer to $c^{\max}$, (b) Gaussian-like distribution with mean farther from $c^{\max}$, (c) uniform distribution, and (d) bimodal distribution.

- Algo.1 is the YDS algorithm, while algo.2 is the p-YDS algorithm,
- performance is the expected energy consumption with the computed speed-schedule,
- y are the standard parameters of a given job-set,
- x are the stochastic parameters of a given job-set, and
- X is the following constraints on the stochastic parameters for each job J_i in the job-set
  - each job has at least one phase: $n_i \geq 1$,
  - the sum of the execution times of the phases match the WCET: $\sum_{j=1,\ldots n_i} c_{ij} = c_i^{\max}$, and
  - The probability of all the phases are bounded: $p_{\min} \leq p_{ij} \leq 1$, for $j \in \{1 \ldots n_i\}$,

where $p_{\min}$ is a given bound on the smallest probability of execution of any phase. As we will see, specifying $p_{\min}$ is necessary, as the competitive ratio increases as $p_{\min}$ decreases. In other words, we do not consider highly improbable (with probability less than some given $p_{\min}$) phases of jobs.

The competitive ratio Ω is a function of the standard parameters of the jobs (in (6) this is represented by y). Ω may also depend on number of jobs, when they arrive, when they have deadlines, and how long they execute in the worst-case. Our aim is to compute Ω as generally as possible, i.e., without dependence on many of these parameters.

4.2 Competitive Ratio for Job-Set with a Single Job

To start our discussion, we begin by considering job-sets with a single job. Since, we consider the relative energy consumption of two algorithms, we can

normalize the deadline, i.e., fix it to some constant. Thus, the only standard parameter is the WCET, $c^{\max}$. The stochastic parameters must characterize the probabilities of different phases of the job, such that the phase with the smallest probability (last phase of the job) has a probability of at least $p_{\min}$. Further, the sum of the execution times of the phases must not exceed $c^{\max}$. With these constraints we can identify innumerable probability distributions of the phases. Some are illustrated in Figure 4. To compute the competitive ratio, we must identify which amongst these variety of distributions leads to the largest ratio $\omega = \overline{E}_{\text{YDS}}/\overline{E}_{\text{p-YDS}}$. Prima facie, this is not clear: A distribution with a high variation in the execution time would reduce the expected energy consumption for the schedules computed by both algorithms.

The first result shows that to maximize ω, there must be only two distinct probabilities of the phases: 1 and $p_{\min}$.

Lemma 5 *Given any job J_i with $p_{ij} \in [p_{\min}, 1] \; \forall \; p_{ij} \in \mathbf{p}_i$. If $\omega = \overline{E}_{YDS}/\overline{E}_{p\text{-}YDS}$ is maximized then the following holds*

$$p_{ij} = p_{\min} \;\vee\; p_{ij} = 1, \quad \forall \; p_{ij} \in \mathbf{p}_i. \tag{7}$$

Proof: Let s_i be the speed assigned by YDS algorithm to execute J_i and let $\overline{s_i}$ be the nominal-speed assigned by p-YDS algorithm. Then

$$s_i = \frac{c_i^{\max}}{d_i - a_i},$$
$$\overline{s_i} = \frac{\sum_{k=\{1,\ldots,n_i\}} c_{ik} \sqrt[\alpha]{p_{ik}}}{d_i - a_i}.$$

Using the above formulas, the analytical expression for ω is

$$\omega = \frac{(c_i^{\max})^{(\alpha-1)} \sum_{k=\{1,\ldots n_i\}} c_{ik} p_{ik}}{\left(\sum_{k=\{1,\ldots n_i\}} c_{ik} \sqrt[\alpha]{p_{ik}}\right)^{\alpha}}. \tag{8}$$

Differentiating ω w.r.t. the probabilities we have

$$\left.\frac{\partial \omega}{\partial p_{ik}}\right|_p \neq 0, \quad \forall \; p \in (1, p_{\min}), \quad \forall \; k \in \{1, \ldots, n_i\}. \tag{9}$$

Thus, the only extrema are found at the two bounds on the probabilities, i.e., at 1 and $p_{\min}$. □

From the above result, it follows that to maximize ω, the probability of the phases must look as shown in Figure 5. Specifically, the probability of execution until some execution time $\widehat{c}$ is 1, and then drops to the minimum allowed probability of $p_{\min}$. In the following result, we compute the value of $\widehat{c}$.

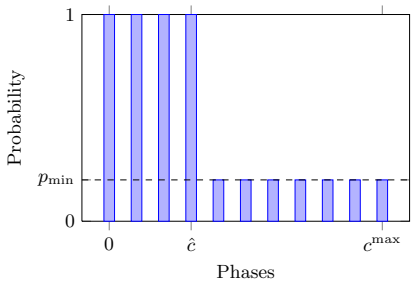

Figure 5: Stochastic parameters which maximize ω. All phases until execution time $\widehat{c}$ have the maximum probability 1, while all subsequent phases have the lowest probability of $p_{\min}$.

Lemma 6 *Given any job J_i of two phases with (a) $p_{i1} = 1$ and $p_{i2} = p_{\min}$, and (b) $c_{i1} + c_{i2} = c_i^{\max}$ for some constant $c_i^{\max}$. Then, the ratio ω is maximized if*

$$c_{i1} = \left(\frac{c_i^{\max}}{\alpha - 1}\right)\left(\frac{\sqrt[\alpha]{p_{\min}}}{1 - \sqrt[\alpha]{p_{\min}}} - \frac{\alpha\ p_{\min}}{1 - p_{\min}}\right), \tag{10}$$

$$c_{i2} = \left(\frac{c_i^{\max}}{\alpha - 1}\right)\left(\frac{\alpha}{1 - p_{\min}} - \frac{1}{1 - \sqrt[\alpha]{p_{\min}}}\right). \tag{11}$$

Proof: Consider the analytical expression of ω in (8). If we set $n_i = 2$, $p_{i1} = 1$ and $p_{i2} = p_{\min}$, we have

$$\omega = \frac{(c_i^{\max})^{(\alpha-1)}\,(c_{i1} + c_{i2} \cdot p_{\min})}{\left(c_{i1} + c_{i2}\sqrt[\alpha]{p_{\min}}\right)^{\alpha}}. \tag{12}$$

We substitute $c_{i2} = c_i^{\max} - c_{i1}$, and differentiate ω w.r.t. c_{i1}. The derivative vanishes when (10) and (11) hold. □ From the above result, we obtain an analytical expression for $\widehat{c}$ as given in (11). Then, by combining the results of the last two lemmas, we directly obtain the competitive ratio.

Theorem 3 *The competitive ratio of the YDS algorithm for job-sets with a single job, denoted Ω_1, is equal to*

$$\left(\frac{\alpha - 1}{\sqrt[\alpha]{p_{\min}} - p_{\min}}\right)^{(\alpha-1)}\left(\frac{1 - p_{\min}}{\alpha}\right)^{\alpha}\left(\frac{1}{1 - \sqrt[\alpha]{p_{\min}}}\right) \tag{13}$$

The competitive ratio is a function of the exponent of speed in power consumption, α, and the minimum probability of a phase $p_{\min}$. This function is plotted in Figure 6. As power consumption becomes a stronger function of the speed, i.e., as α increases, Ω_1 also increases. Further, as longer executions of the job become less probable, i.e., as $p_{\min}$ decreases, Ω_1 increases. For $\alpha = 2.5$ and $p_{\min} = 0.25$, we have $\Omega_1 = 1.152$, i.e., the expected energy of the speed-schedule computed by the YDS algorithm can be up to 15% higher than the optimal.

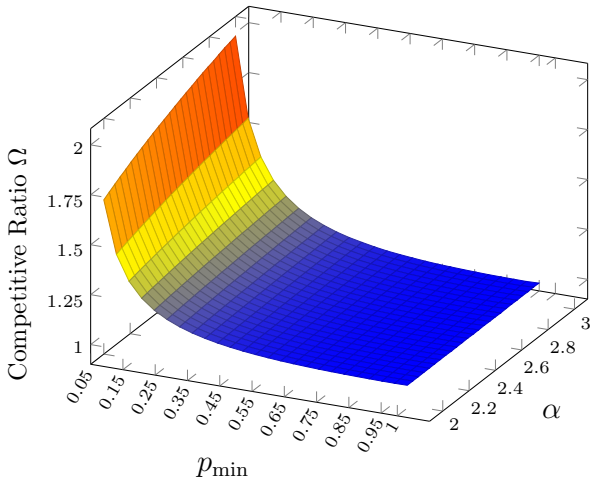

Figure 6: Competitive ratio of the YDS algorithm w.r.t. the p-YDS algorithm, as a function of the exponent of speed in the power function α, and the minimum probability of execution of a phase $p_{\min}$.

4.3 Competitive Ratio of Any Job-Set

Recall that Ω is a function of the standard parameters. In the previous part, we focused on job-sets with a single job. Now we consider any job-set, with given number of jobs, arrival times, deadlines and WCETs. Significantly, we show in the following result that we can compute an exact competitive ratio for any such job-set. Furthermore, this competitive ratio is the same as the competitive ratio of a single job Ω_1 computed in Theorem 3.

Theorem 4 *The competitive ratio of the YDS algorithm for any job-set, denoted Ω_*, is equal to Ω_1 as derived in (13).*

Proof: We need to show two different things. One, for any given job-set we can assign the stochastic parameters such that $\omega = \Omega_1$, i.e., prove that Ω_1 is a lower bound on Ω_*. Two, for any given job-set we cannot assign the stochastic parameters such that $\omega > \Omega_1$, i.e., prove that Ω_1 is an upper bound on Ω_*.

Lower bound on Ω_*: For every job $J_i \in \mathcal{J}$, we set the parameters of phases as given in Lemma 6, i.e., there are two phases with probabilities 1 and $p_{\min}$ with execution time as given in (10) and (11). Then, for any interval I, the nominal-speed computed in p-YDS algorithm, s_I, is given as

$$s_I = \gamma \frac{\sum_{J_i \in \mathcal{J}_\mathcal{I}} c_i^{\max}}{|I|}, \tag{14}$$

where $\mathcal{J}_\mathcal{I}$ is the set of jobs with arrival times and deadlines within I, and γ is some constant depending only on α and $p_{\min}$. Indeed, s_I is proportional to the speed that YDS algorithm assigns to an interval (above expression without γ).

	Standard Params.			Stochastic Params.		
J_i	a_i	d_i	$c_i^{\max}$	n_i	(c_{i1}, p_{i1})	(c_{i2}, p_{i2})
J_1	0	8	6	2	(0.86, 1)	(5.14, 1/64)
J_2	5	16	7	2	(1, 1)	(6, 1/64)
J_3	15	25	9	2	(1.29, 1)	(7.71, 1/64)

Table 2: Modified job-set of Example 1. Stochastic parameters are set according to Theorem 3.

Hence, the order in which intervals are assigned speeds is the same in the two algorithms. Furthermore, in any interval, the time for which each job executes is the same for both algorithms, as c_{i1} and c_{i2} are proportional to $c_i^{\max}$. Thus, every job $J_i \in \mathcal{J}$ executes within the same interval in the schedules generated by YDS and p-YDS algorithms. Then, from Theorem 4, the value of ω is equal to Ω_1.

Upper bound on Ω_*: We prove this by contradiction. Let for some job-set $\omega = \overline{E}_{\text{YDS}}/\overline{E}_{\text{p-YDS}} > \Omega_1$. Consider a different speed-schedule $\tilde{S}$ such that it (a) assigns a single speed for each job, and (b) assigns the same execution window for each job as in the speed-schedule computed by p-YDS algorithm. From (a) and the optimality of the YDS algorithm we know that $\overline{E}_{\text{YDS}} \leq \overline{E}_{\tilde{S}}$. From (b) and the computation of Ω_1, we know that $\overline{E}_{\tilde{S}}/\overline{E}_{\text{p-YDS}} = \Omega_1$. This contradicts the assumption that $\omega > \Omega_1$. □

Illustrating Example We revisit the example from the previous section. For the job parameters as given in Table 1, we obtained $\omega = 2.06$. In this example $\alpha = 3$ and $p_{\min} = 1/64$. For these values we have $\Omega_1 = 3.43$. From Theorem 4, we know that it is possible to modify only the stochastic parameters such that we increase the observed ω to 3.43.

Consider the job parameters shown in Table 2. Here we divide each job into two phases with the probabilities 1 and 1/64. The execution times of the phases are as in (10) and (11). For this case the two generated speed-schedules are shown in Figure 7. The expected energy consumption is $\overline{E}_{\text{YDS}} = 2.67$ and $\overline{E}_{\text{p-YDS}} = 0.78$. The value of $\omega = 3.43$ matches that computed as Ω_1. This illustrates the main result of this section: For a given job-set we can choose the stochastic parameters such that the ratio ω equals Ω_1.

5 p-YDS Algorithm for a Given Upper-Bound on the Speed

Apart from minimizing the worst-case energy consumption, the YDS algorithm also optimally minimizes the highest speed used in the speed-schedule. In contrast, the p-YDS algorithm achieves a smaller expected energy by executing phases of lower probabilities with higher speeds. In the limit, the highest speed

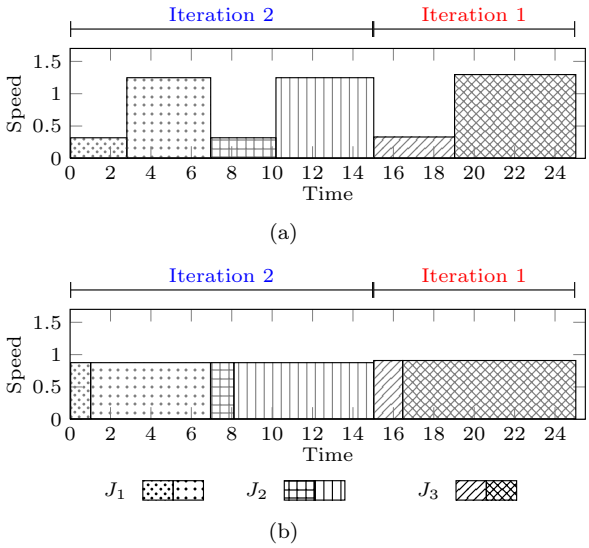

Figure 7: Speed-schedules for the job-set of Table 2 computed by (a) p-YDS and (b) YDS algorithms. The expectation of energy $\overline{E}$ for the two schedules are (a) 0.78 and (b) 2.67, with $\omega = 3.43$.

used by p-YDS algorithm can be a factor $(1/\sqrt[\alpha]{p_{\min}})$ larger than that of the YDS algorithm. Such high speeds may not be available on some processors. In this section, we optimally extend p-YDS algorithm to consider a given highest speed, denoted $s_{\max}$. Proof of results in this section are in the appendix.

5.1 Job-Sets with a Single Job

Using Karush–Kuhn–Tucker (KKT) conditions of optimality of non-linear programs, we relate the speeds of the different phases, for a job-set with a single job.

Lemma 7 *The optimal speed-schedule for a single job J_i, for a given highest speed $s_{\max}$, must satisfy*

$$s_{ik} = s_{\max} \Rightarrow s_{ij} = s_{\max}, \qquad \forall\ j,k,\ \ s.t.\ p_{ij} \leq p_{ik}, \tag{15}$$

$$\frac{s_{ij}}{s_{ik}} = \sqrt[\alpha]{\frac{p_{ik}}{p_{ij}}}, \qquad \forall\ j,k,\ \text{s.t.}\ s_{ij}, s_{ik} < s_{\max}. \tag{16}$$

This result can be considered as an extension of Lemma 3, to the case of a given highest speed. It states that (a) a (possibly empty) sub-set of the phases, with the lowest probability of execution, are assigned the highest speed $s_{\max}$, and (b) the speeds of the remaining phases satisfy the earlier condition from Lemma 3.

Using this result, we present an algorithm to iteratively compute the speeds in the following. A similar approach, without proof of optimality, was proposed in [4].

Description 1 (Iterative Speed Bounding) *Given is any job and its standard and stochastic parameters. We compute the speed for each phase using Lemmas 2 and 3. If the speed of a phase is larger than $s_{\max}$, then its speed is reduced to $s_{\max}$, and the speeds for the remaining phases are recomputed using Lemmas 2 and 3. This is repeated until there are no phases with speeds larger than $s_{\max}$.*

We formally describe the algorithm in the appendix. We prove optimality in the following result, and subsequently illustrate with an example.

Theorem 5 *For a single job and a given highest speed, the expected energy consumption is minimal with the speed-schedule computed by iterative speed bounding.*

Illustrating Example

We compute the nominal-speed of jobs J_2 and J_3 of the job-set of Table 1 to execute within the interval $[8, 25]$ with $s_{\max} = 1.25$. Later on, we will use this to generate the speed-schedule shown in Figure 8(a). Using Lemma 3, we obtain $\overline{s} = 0.412$. With this speed, phase 3 of J_2 has a speed (1.648) higher than $s_{\max} = 1.25$. We set $s_{23} = 1.25$, and repeat the computation. Again, using Lemma 3, we obtain $\overline{s} = 0.435$. With this speed, phase 3 of J_3 has a speed (1.304) greater than 1.25. We set $s_{33} = 1.25$, and repeat the computation. Again, using Lemma 3, we obtain $\overline{s} = 0.444$. Now, no phase has a speed greater than 1.25. Hence, we end with $\overline{s_2} = \overline{s_3} = 0.444$.

5.2 Job-Sets with Multiple Jobs

For job-sets with multiple jobs, we can apply the iterative bounding approach to compute the nominal-speed for an interval. With this change, we now show that the interference condition of the optimal speed-schedule of Theorem 1 holds, i.e., an interfering job must have a higher nominal-speed than the interfered job.

Theorem 6 *Consider any two jobs J_i, J_j. For the optimal speed-schedule for a given highest speed, if J_i interferes with J_j, then the nominal-speeds of the two jobs computed using iterative speed bounding must satisfy $\overline{s_i} \geq \overline{s_j}$.*

The above necessary condition is the main enabler to compute the optimal speed-schedule for arbitrary job-sets. With this result, as in the original p-YDS algorithm, we extend the optimality result of Theorem 2 for a given highest speed.

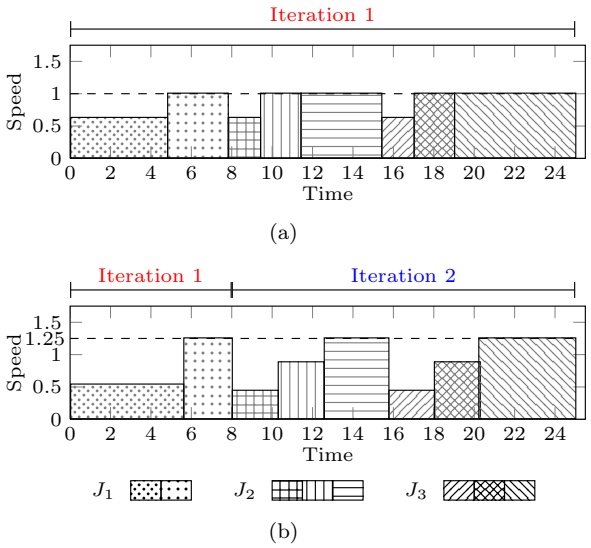

Figure 8: Speed-schedules for the job-set of Table 1 computed by p-YDS algorithm for (a) $s_{\max} = 1$, and (b) $s_{\max} = 1.25$. The change in $s_{\max}$ leads to different intervals and speed-schedules.

Illustrating Example

Consider the job-set of Table 1. We compute the schedules with the p-YDS algorithm for two highest speeds: 1 and 1.25. In both cases, feasible schedules exist, as the highest speed used in the speed schedule computed by the YDS algorithm is 1 (Figure 3(b)). The obtained schedules and the chosen intervals in each iteration of the algorithm are shown in Figure 8. Notice that the schedule for the different highest speeds are different, even in the order in which intervals are chosen. Clearly, as we decrease the given highest speed the optimal expected energy consumption increases: It is 2.19 for no highest speed, 2.269 for $s_{\max} = 1.25$, and 2.855 for $s_{\max} = 1$. With $s_{\max} = 1$, which is the highest speed used by the YDS algorithm (Figure 3(b)), the ratio $\omega = 1.58$.

6 Experimental Results

6.1 Computing ω for Different Tasks

The ratio $\omega = \overline{E}_{\text{YDS}}/\overline{E}_{\text{p-YDS}}$ specifies the relative energy cost of being agnostic to variability in the execution time. We compute this for some real tasks, considering a job-set with a single job. As discussed, this depends only on the `CDF` of the execution demand of these tasks. In the appendix, we present the `CDF` of certain multimedia and encryption tasks sourced from [3] Based on these, the derived values of ω are shown in Table 3, for a chosen value of $\alpha = 2.5$. On

h264	blowfish	tmn	tmndec	madplay	mgplay	toast
1.166	1.131	1.059	1.193	1.074	1.159	1.189

Table 3: ω for profiled tasks executing as single jobs

the high end, task `tmndec` has $\omega = 1.193$. Thus, the expected energy is about 20% higher than the optimal when using the YDS algorithm to optimize speeds. On average, we obtain $\omega = 1.138$. Putting this in (13), for $\alpha = 2.5$, we obtain $p_{\min} = 0.12$. In other words, on average the tasks have the same energy cost as with worst-case stochastic parameters for $p_{\min} = 0.12$.

6.2 Energy Ratios for a Given Job-Set

In the previous experiment, we studied the ratio ω for a given set of tasks executing as single jobs. Now we study the ratio ω when these jobs are scheduled together, i.e., job-sets with multiple jobs. Indeed, in any such job-set ω would depend on quantities like arrival times, deadlines, and execution demands. But can something be said of the ω independent of these quantities? Indeed we can show that resultant ω is bounded as given in the following result. The proof is provided in the appendix.

Theorem 7 *For a given set of jobs $\mathcal{J}$, let ω_i denote the ratio of the expected energy for the job J_i when executed as a single job. Then, for the job-set $\mathcal{J}$, the resultant ω satisfies*

$$\omega \leq \max_{J_i \in \mathcal{J}} \omega_i \tag{17}$$

As an example, consider a job-set of two jobs, the `tmn` and `tmndec`. The execution demands of both are normalized to 1. They both arrive at time 0. The `tmn` job has a deadline of 5, while the deadline of the `tmndec` job is varied. For different values of this deadline, the obtained values of ω are shown in Figure 9. As can be seen the resultant ω varies between the individual ω of the two tasks computed before. When the deadline is very small, the `tmn` task is a significant contributor to the energy and thus the resultant ω tends to $\omega_{\texttt{tmn}}$. Conversely, when the deadline is very large, `tmn` contributes much less energy, and ω tends to $\omega_{\texttt{tmndec}}$.

To summarize, we can profile the `CDF` of individual tasks and compute their respective ω values. Then any job-set comprising of these tasks has a resultant ω that is a upper-bounded by the individual jobs' ω, independent of arrival times, deadlines, and relative execution times. This enables one to test individually for each job the potential energy cost of not considering the variation in the execution times.

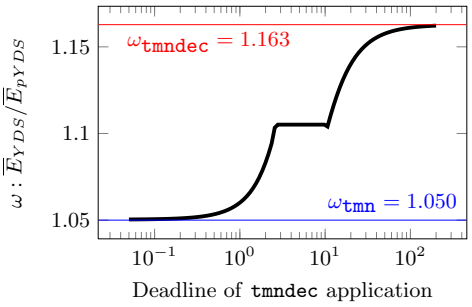

Figure 9: Variation in the energy ratio ω as a function of the deadline of one the tasks. Here ω varies between the ω computed for the individual tasks.

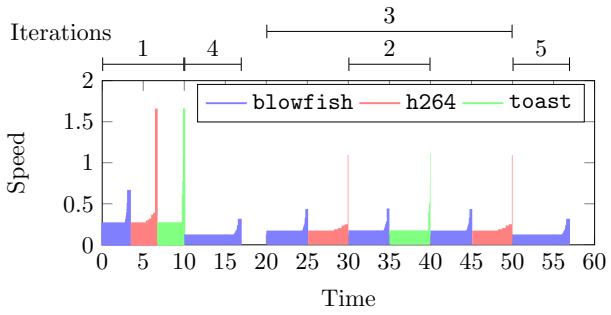

Figure 10: The speed schedule computed by p-YDS algorithm for a periodic task-set with three tasks within the hyper-period. Different jobs of the same task execute at different speeds.

6.3 Example Schedule of Periodic Tasks

The p-YDS algorithm works with individual jobs. It can thus consider periodic tasks, where the schedule is computed for a regular window, such as the hyperperiod. We illustrate this with a periodic task-set of three tasks, namely `blowfish`, `h264`, and `toast`. In the same order, the periods are 10, 20, and 30, and the relative deadlines are 7, 10, and 10. Consider a hyperperiod of length 60, starting from time 0 when jobs of all tasks arrive. The speed-schedule computed by p-YDS algorithm is shown in Figure 10. Interestingly, different jobs of the same task are executed with different speeds. For instance, the first 4 jobs of the `blowfish` task belong to different iterations in the algorithm and execute at 4 different nominal speeds.

7 Conclusions

Meeting hard real-time guarantees could be considered in conjunction with other objectives which may not be worst-case. In this work, we considered the expected energy consumption. This was motivated from the uncertainty and variability in the execution demand of jobs. The presented p-YDS algorithm extended the well known YDS algorithm. The competitive ratio compared the two algorithms in terms of the expected energy. Significantly, we showed that this ratio is independent of the parameters of the job-set. Further, for a given set of specific tasks, the ratio of the expected energy for the two algorithms is bounded by the maximum ratios for the individual tasks.

References

[1] F. F. Yao, A. J. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *FOCS*, pp. 374–382, IEEE Computer Society, 1995.

[2] N. Bansal and K. Pruhs, "Speed scaling to manage temperature," in *STACS 2005*, pp. 460–471, Springer, 2005.

[3] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, (New York, NY, USA), pp. 149–163, ACM, 2003.

[4] C. Xian, Y.-H. Lu, and Z. Li, "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1467–1478, 2008.

[5] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *SIGMETRICS/Performance*, pp. 50–61, ACM, 2001.

[6] J. E. Fritts, F. W. Steiling, and J. A. Tucek, "Mediabench ii video: Expediting the next generation of video systems research," in *Electronic Imaging 2005*, pp. 79–93, International Society for Optics and Photonics, 2005.

[7] D. Mosse, H. Aydin, B. Childers, and R. Melhem, "Compiler-assisted dynamic power-aware scheduling for real-time applications," in *In Workshop on Compilers and Operating Systems for Low Power*, 2000.

[8] F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and dvs processors," in *ISLPED* (E. Macii, V. De, and M. J. Irwin, eds.), pp. 46–51, ACM, 2001.

[9] R. Xu, D. Mossé, and R. G. Melhem, "Minimizing expected energy in real-time embedded systems," in *EMSOFT* (W. Wolf, ed.), pp. 251–254, ACM, 2005.

[10] R. Xu, R. Melhem, and D. Mossé, "A unified practical approach to stochastic dvs scheduling," in *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, pp. 37–46, ACM, 2007.

[11] J.-J. Chen and L. Thiele, "Expected system energy consumption minimization in leakage-aware dvs systems," in *ISLPED*, pp. 315–320, ACM, 2008.

[12] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *ACM SIGOPS Operating Systems Review*, vol. 35, pp. 89–102, ACM, 2001.

[13] S. Saewong and R. Rajkumar, "Practical voltage-scaling for fixed-priority rt-systems," in *Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings. The 9th IEEE*, pp. 106–114, IEEE, 2003.

Appendix

Proof: [of Lemma 1] Let some schedule that meets all deadlines have the minimal expected energy consumption. Let its scheduling policy not coincide with EDF. Consider modifying the schedule, where each phase of each job executes at the same speed, whereas the ordering of the jobs is changed to EDF. As EDF optimally minimizes the lateness, we know that this changed schedule also meets all deadlines. Further, as the speeds are unchanged, the expected energy consumption is also minimal. Thus, we can always design an optimal speed schedule that follows the EDF policy. □

Proof: [of Lemma 2] For any convex function we have $\forall \lambda \in [0, 1]$

$$\lambda f(a) + (1 - \lambda) f(b) \geq f(\lambda a + (1 - \lambda) b). \tag{18}$$

Let a given phase execute at two different speeds a and b for times that are proportional to λ and $(1 - \lambda)$ respectively. We can alternatively execute the phase at speed $(\lambda a + (1 - \lambda)b)$ with the same finish time. Let f denotes the convex function that is the power consumption at a particular speed. Then applying the above definition of convexity we have the result. □

Proof: [of Lemma 3] We prove this by applying the Karush-Kuhn-Tucker conditions on the following optimization program.

$$\begin{aligned} \text{minimize} \sum_{k=1}^{N} p_{ik} \times c_{ik} \times (s_{ik})^{\alpha - 1} \\ \text{s.t.} \sum_{i=1}^{N} \frac{c_{ik}}{s_{ik}} \leq d_i - a_i. \end{aligned}$$

Clearly, the energy consumption is minimized when the constraint is tight. Then, from the KKT conditions, for some constant λ, we have

$$\nabla (p_{ik} c_{ik} s_{ik}^{(\alpha - 1)}) + \lambda \nabla \left(\sum_{k=1,2,\ldots,n_i} \frac{c_{ik}}{s_{ik}} \right) = 0.$$

Taking partial derivatives, w.r.t. s_{ik} we have

$$\frac{\lambda}{\alpha - 1} = p_{ik} s_{ik}^{\alpha}$$

As the left hand side in the above equation is a constant, we have $s_{ik} \propto \sqrt[\alpha]{p_{ik}}$, for all phases of the job. □

Proof: [of Lemma 4] We prove this by contradiction for some two consecutive iterations of p-YDS algorithm. Let I_{n-1} and I_n be the two intervals assigned the nominal-speed in iterations $n-1$ and n, respectively. Let the nominal-speed assigned in iteration n be higher than that in iteration $n-1$.
Case (a): The two intervals are non-intersecting. In this case, the nominal-speed to be assigned to I_n must remain the same in the two iterations. But we know that, in iteration $n-1$, the nominal-sped is highest for interval I_n. This contradicts the assumption.
Case (b): The two intervals are intersecting. In this case, consider the interval $I = I_{n-1} \cup I_n$. We can show that I has a higher nominal-speed than I_{n-1} in iteration $(n-1)$. This again contradicts the assumption. □

Algorithm 2: Iterative Speed Bounding

Input: A job J_i with its standard and stochastic models
Input: The maximum available speed $s_{\max}$
Output: The speed of all phases of J to minimize the expected energy

$\Delta \leftarrow d_i - a_i$
$\mathcal{P} \leftarrow \{1, 2, \ldots, n_i\}$
repeat
  changed $\leftarrow$ false
  $\overline{s_i} = \frac{\sum_{k \in \mathcal{P}} c_{ik} \sqrt[\alpha]{p_{ik}}}{\Delta}$
  foreach $k \in \mathcal{P}$ **do**
    $f_{ik} \leftarrow \overline{s_i} / \sqrt[\alpha]{p_{ik}}$
    if $f_{ik} > s_{\max}$ **then**
      $f_{ik} \leftarrow s_{\max}$
      $\mathcal{P} \leftarrow \mathcal{P} - \{k\}$
      $\Delta \leftarrow \Delta - (c_{ik}/s_{\max})$
      changed $\leftarrow$ true
until $\neg$*changed*

Proof: [of Lemma 7] Both these are shown using the KKT optimality conditions for the following non-linear program

$$\text{minimize} \quad \sum_{k=1,2,\ldots,n_i} p_{ik} c_{ik} s_{ik}^{(\alpha-1)} \tag{19}$$

$$\text{subject to} \quad \sum_{k=1,2,\ldots,n_i} \frac{c_{ik}}{s_{ik}} \leq d_i - a_i, \tag{20}$$

$$s_{ik} \leq s_{\max}, \forall\ k = \{1, 2, \ldots, n_i\}. \tag{21}$$

Let the non-negative Lagrangian multipliers for inequality (20) be λ, and for the set of inequalities (21) be λ_k for the kth phase. The multipliers corresponding to non-tight inequalities are 0. Clearly, (20) is tight. Let σ be the set of phases for which (21) is tight. Then, according to the KKT conditions,

$$\nabla(p_{ik} c_{ik} s_{ik}^{(\alpha-1)}) + \lambda \nabla \left(\sum_{k=1,2,\ldots,n_i} \frac{c_{ik}}{s_{ik}} \right) + \sum_{k \in \sigma} \lambda_k \nabla s_{ik} = 0.$$

Taking partial derivatives w.r.t. s_{ik} for $k \notin \sigma$, i.e., $s_{ik} < s_{\max}$, we get

$$\lambda = (\alpha - 1) p_{ik} s_{ik}^{\alpha}. \tag{22}$$

This proves (16). Taking partial derivatives w.r.t. s_{ik} for $k \in \sigma$, i.e., $s_{ik} = s_{\max}$, we get

$$\lambda = (\alpha - 1) p_{ik} s_{ik}^{\alpha} + \frac{\lambda_k}{c_{ik}}. \tag{23}$$

As $\lambda_k > 0$, combining (22) and (23), we get (15). □

Proof: [of Theorem 5] The speed-schedule computed by the iterative speed bounding satisfies (15) as this is enforced by the algorithm in Algorithm 1. As speed assigned to a less probable phase is higher, in any iteration, a phase will be assigned the speed $s_{\max}$ only after all lesser probable phases are assigned $s_{\max}$. This satisfies (16). Finally, the computed schedule utilizes the full interval in each iteration. □

Proof: [of Theorem 6] We follow the same approach as in the proof of Theorem 1. We prove this by contradiction. For the two defined jobs J_i and J_j, let $\overline{s_i} > \overline{s_j}$ for some speed schedule $\mathcal{S}$. Let $\mathcal{P}_i, \mathcal{P}_j$ denote the set of phases of J_i and J_j which are assigned the speed $s_{\max}$ in $\mathcal{S}$. Let $\overline{\mathcal{P}_i}, \overline{\mathcal{P}_j}$ denote the respective complements. Let E_i, E_j denote the expected energy consumption for any job J_i and J_j. Then E_i (and similarly E_j) is given as

$$E_i = \sum_{k \in \overline{\mathcal{P}_i}} p_{ik} \cdot c_{ik} \cdot \left(\frac{\overline{s_i}}{\sqrt[\alpha]{p_{ik}}} \right)^{(\alpha-1)} + \sum_{k \in \mathcal{P}_i} p_{ik} c_{ik} s_{\max}^{(\alpha-1)}.$$

Since, we do not want to change the schedule for all other jobs, the total time for which J_i and J_j execute must be a constant, i.e.,

$$\sum_{l=\{i,j\}} \left(\sum_{k \in \overline{\mathcal{P}_l}} \frac{c_{lk}}{\overline{s_l} / \sqrt[\alpha]{p_{lk}}} + \sum_{k \in \mathcal{P}_l} \frac{c_{lk}}{s_{\max}} \right) = \text{constant}.$$

The derivative of $E_i + E_j$ with respect to s_i while satisfying the above constraint is given as

$$\frac{d(E_i + E_j)}{d\overline{s_i}} = \eta \cdot (s_j^{(\alpha-2)} - s_i^{(\alpha-2)}),$$

where η is a positive constant. For $\alpha \geq 2$ and $\overline{s_i} > \overline{s_j}$, the above quantity is negative. Thus, we can reduce $\overline{s_i}$ and correspondingly increase $\overline{s_j}$ until either (a) $\overline{s_i} = \overline{s_j}$, or (b) J_j does not execute within $[a_i, d_i]$. □

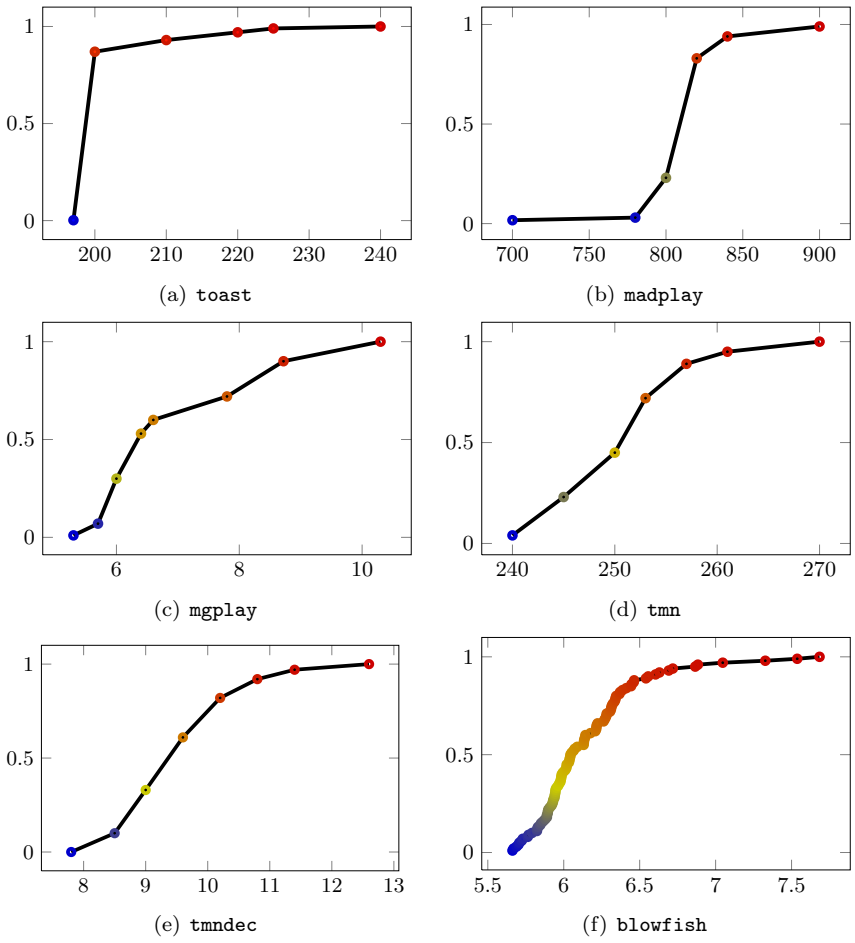

Figure 11: CDF for different considered applications.

Proof: [of Theorem 7] For a given job-set, let S denote a speed-schedule wherein every job is executed within the same interval as in the optimal schedule computed by the p-YDS algorithm. However, S uses a single speed per job. Within each such interval, the ratio of expected energy consumed in S and S_{p-YDS} is exactly equal to the ω computed for the corresponding task. Then, the ratio of the total expected energy $\overline{E}_S/\overline{E}_{\text{p-YDS}}$ is bounded by the maximum of the ω for the individual tasks. Also, from the optimality of the YDS algorithm, we have $\overline{E} \leq \overline{E}_{\text{YDS}}$. Thus, $\overline{E}_{\text{YDS}}/\overline{E}_{\text{p-YDS}}$ is upper-bounded by the maximum ω of the individual tasks. □