Robust Online Speed Scaling With Deadline Uncertainty

Goonwanth Reddy
Department of Electrical Engineering,
Indian Institute of Technology, Madras, Chennai, India
ngoonwanth@gmail.com

Rahul Vaze
School of Technology and Computer Science,
Tata Institute of Fundamental Research, Mumbai, India
vaze@tcs.tifr.res.in

Abstract

A speed scaling problem is considered, where time is divided into slots, and jobs with payoff v arrive at the beginning of the slot with associated deadlines d. Each job takes one slot to be processed, and multiple jobs can be processed by the server in each slot with energy cost $g(k)$ for processing k jobs in one slot. The payoff is accrued by the algorithm only if the job is processed by its deadline. We consider a robust version of this speed scaling problem, where a job on its arrival reveals its payoff v, however, the deadline is hidden to the online algorithm, which could potentially be chosen adversarially and known to the optimal offline algorithm. The objective is to derive a robust (to deadlines) and optimal online algorithm that achieves the best competitive ratio. We propose an algorithm (called min-LCR) and show that it is an optimal online algorithm for any convex energy cost function $g(.)$. We do so without actually evaluating the optimal competitive ratio, and give a general proof that works for any convex g, which is rather novel. For the popular choice of energy cost function $g(k) = k^\alpha, \alpha \geq 2$, we give concrete bounds on the competitive ratio of the algorithm, which ranges between 2.618 and 3 depending on the value of α. The best known online algorithm for the same problem, but where deadlines are revealed to the online algorithm has competitive ratio of 2 and a lower bound of $\sqrt{2}$. Thus, importantly, lack of deadline knowledge does not make the problem degenerate, and the effect of deadline information on the optimal competitive ratio is limited.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Online Algorithms, Speed Scaling, Greedy Algorithms, Scheduling

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2018.22

1 Introduction

Energy efficient transmission of packets in communication systems or processing of jobs in microprocessors (and similar applications) is a fundamental resource allocation problem. A job/packet can be processed/transmitted by a server 'fast' but only at the cost of higher energy consumption. Typically, the energy cost is a convex function of the server speed, (e.g., a popular choice is x^α for $\alpha \geq 2$) and the general objective is two fold: maximize the profit from processing jobs while incurring minimum energy cost. The profit can either be the payoff accrued on processing a job or some function of the inverse of the processing time. This problem is known as *speed scaling* problem in literature, where the speed of the server is the tunable parameter which determines the profit and the energy consumption.

Speed scaling problem has been considered with both the infinite speed model as well as bounded speed model, where in the former case, the server is allowed to scale its speed

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018).
Editors: Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer; Article No. 22; pp. 22:1–22:17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

anywhere in $[0, \infty)$, while in the latter case, it is bounded by a fixed constant. Infinite speed model allows processing of all jobs, and the main concern is to minimize energy consumption, while in the bounded speed model, both maximizing the profit and minimizing the energy consumption is a challenge.

Another broad classification considered with the speed scaling problem is with respect to deadlines. In the case when jobs do not have deadlines, a typical objective is to minimize a linear combination of the sum of the flow time (completion time minus the arrival time) and the energy consumption for each job. When jobs also specify a deadline, under the bounded speed model, the objective is to process jobs that maximize the profit (accrued only from jobs that are processed by their deadline) while minimizing the energy cost.

An alternate speed scaling model (which we consider in this paper) is a discretized one, where time is divided into slots, and set of jobs S with payoff $v_i, i \in S$ arrive at the beginning of the slot with associated deadlines $d_i, i \in S$. Each job takes one slot to be processed, and multiple jobs can be processed by the server in each slot with energy cost $g(k)$ for processing k jobs in one slot. The payoff is accrued by the algorithm only if the job is processed by its deadline, and the objective is to maximize the sum of the profit (payoff minus energy cost) of the processed jobs.

One limiting aspect of almost all literature on speed scaling with job deadlines is the need for the exact knowledge of deadlines. The derived results critically depend on exact deadline information, and are not robust to even a small uncertainty. In modern applications, the job deadlines could be time varying, could potentially depend on other jobs or their completion times, or may not be precisely known on the job arrival. Towards addressing this critical aspect as well as to generalize the model, we consider a robust version of the discretized speed scaling problem, where a job on its arrival reveals its payoff v, however, the deadline is hidden to the online algorithm, which could potentially be chosen adversarially and known to offline optimal algorithm. This approach will lead to the derivation of robust online algorithms for speed scaling and provides a means to quantify the fundamental effect of deadline knowledge on speed scaling. We note that the robust approach is useful only if the problem itself does not become degenerate, in the sense that no online algorithm can achieve a reasonable competitive ratio.

For the considered robust model, we propose a simple online algorithm, called the minimum-local competitive ratio (min-LCR) algorithm, that does not need any deadline information. Let $c_k = g(k) - g(k-1)$ be the effective energy cost for processing the k^{th} job. Algorithm min-LCR indexes all the available jobs at each slot in the non-increasing order of their payoffs v, and computes the profit p_ℓ it will make if it processes $1 \leq \ell \leq m$ jobs, where m is such that $v_m - c_m > 0$ and $v_{m+1} - c_{m+1} < 0$. It also presumes a worst case scenario for the deadlines (since it does not know the exact deadlines) where the ℓ-chosen jobs for processing have deadlines infinity while the left-over jobs (other than the ℓ chosen jobs) have deadlines that expire in the current slot. Let o_ℓ be the profit that can be made by an offline algorithm under the knowledge of the worst choice of deadlines for the current set of available jobs, assuming no further jobs arrive. We define $\mathsf{LCR}_\ell = \frac{o_\ell}{p_\ell}$ for $1 \leq \ell \leq m$, which has the interpretation of local competitive ratio (ratio of the optimal offline and the simple online algorithm under the worst case deadlines given no further jobs arrive), and the algorithm chooses to process ℓ jobs for which LCR_ℓ is minimum. Note that always choosing $\ell = m$ will correspond to a natural greedy algorithm for this problem.

1.1 Contributions

We make the following contributions in this paper.

- We show that the proposed Algorithm min-LCR is an optimal online algorithm for any convex energy cost function $g(.)$. We do so without actually finding the optimal competitive ratio, which is fundamentally different than the typical proof strategy used in the analysis of online algorithms. To the best of our knowledge, such a result is not available in the speed scaling literature to achieve the optimal competitive ratio even when deadlines are exactly known.
- The optimality of the min-LCR algorithm is shown via first constructing a lower bound which with deadline uncertainty is easier compared to prior work when deadlines are revealed, since deadlines can be chosen arbitrarily that are not revealed to any online algorithm. The more challenging task is to show that the proposed min-LCR algorithm actually achieves the lower bound even when it works without any knowledge about the deadlines of any of the jobs.
- To ensure that the considered robust speed scaling problem is not degenerate in the sense that the adversarial deadlines choices that are unknown to the online algorithm make the competitive ratio of any online algorithm arbitrarily large, we provide concrete analysis for the most popular energy cost function of $g(k) = k^\alpha$. The min-LCR algorithm involves finding the optimal number of jobs to process among the available ones that minimizes the LCR which can be computationally extensive. To simplify the complexity, we also consider a simplified version of the min-LCR algorithm, where exactly $k = \lfloor \beta m \rfloor$ or $k = \lceil \beta m \rceil$ jobs are processed in each slot depending on whichever one has lower LCR, where β is the solution of the equation $x^\alpha + x^{\alpha-1} - 1 = 0$. We also consider the natural greedy special case of min-LCR that always processes m jobs, where m has been defined in the min-LCR algorithm description as above.
- We show that the optimal competitive ratio for energy cost function of $g(k) = k^\alpha$ with $\alpha = 2$ is $\phi + 1$ ($\phi = 1/\delta, \delta = \frac{\sqrt{5}-1}{2}$), and is achieved by the simplified min-LCR algorithm. In comparison, when deadlines are known to the online algorithm, for $\alpha \geq 2$ the best known online algorithm has competitive ratio of 2 and the best known lower bound is $\sqrt{2}$.
- For $\alpha \geq 2.5$, the competitive ratio of the simplified min-LCR algorithm is at most $\phi + 1$, and for any $\alpha \geq 2$, the competitive ratio of the greedy algorithm (min-LCR algorithm with $\ell = m$ always) is at most 3. We also derive a lower bound on the competitive ratio of $\sqrt{2} + 1$ for all $\alpha \geq 2$.
- Thus, we show that for the energy cost function of $g(k) = k^\alpha$, lack of deadline knowledge reduces the optimal competitive ratio by a factor of at most $3/\sqrt{2}$. Thus, the loss in performance because of deadline uncertainty is limited, and precludes the possibility that the considered robust model is inherently weak, and the power of any online algorithm is seriously limited. Moreover, it also obviates the need to study the unknown but stochastic deadline case, since the loss in competitive ratio from fully known to completely adverarial is very small.

1.2 Related Work

Starting with [19], there has been a long line of work on optimal speed scaling for servers with unbounded speed. For energy cost $g(k) = k^\alpha, \alpha > 1$, a $2^{\alpha-1}\alpha^\alpha$-competitive algorithm was derived in [19], whose competitive ratio was subsequently improved upon in [6] to $2\left(\frac{\alpha}{\alpha-1}\right)^\alpha \exp(1)^\alpha$, and eventually in [5] to $4^\alpha/(2\sqrt{\exp(1)\alpha})$.

For the bounded speed case, where all jobs cannot be processed, [10] first derived an online algorithm that is 14-competitive algorithm for throughput (number of processed jobs) and $\alpha^\alpha + \alpha^2 4^\alpha$-competitive for energy. Subsequently, the throughput competitiveness was improved to 4 in [4], which is also the best possible [9].

In addition to speed scaling, an additional feature of *sleeping* was introduced in [15], where all jobs can be processed with $(2^{2\alpha-2}\alpha^\alpha + 2^{\alpha-1} + 2)$-competitiveness in terms of energy, which was improved upon in [14] to get $\alpha^\alpha + 2$-competitiveness in terms of energy under the infinite speed model when all jobs can be processed, and 4-competitive algorithm for throughput (number of processed jobs) and $(\alpha^\alpha + \alpha^2 4^\alpha + 2)$-competitive for energy in the bounded server model.

The no-deadline model where the objective function is to minimize the flow time plus the energy has been considered widely [16, 18, 7], with the most general result obtained in [7] that gives a constant competitive algorithm for all energy cost functions. The speed scaling problem in the no-deadline model, where some information about jobs (either value/weight/density) is hidden is called the non-clairvoyant setting and has been addressed in literature starting with [11] and followed up in [2]. Speed scaling with multiple processors has also been considered in [1] and with non-clairvoyant setting in [13], while modern applications for speed scaling are being addressed in [8], where energy is derived from solar cells for renewable energy harvesting.

The discrete model studied in this paper was first considered in [12], where jobs deadlines are known to the online algorithm, which proposed an online algorithm that is 2-competitive, using the ideas from online request matching [17]. An associated lower bound (easy to construct) on the competitive ratio for this problem is $\sqrt{2}$.

As far as we know the speed scaling problem when jobs have hard deadlines that are not exactly known to the online algorithm has not been considered in literature. In load balancing literature, unknown job deadline case is referred to as scheduling for temporary tasks, where the duration for which a job lasts is unknown [3]. In load-balancing, however, each job has to be scheduled as soon as it arrives, and the only decision variables are : which server to be assigned for each job and the dynamic server speed.

2 Problem Definition

We consider a discrete time system, where time is divided in discrete slots. A sequence $\boldsymbol{\sigma} = J_1, \ldots, J_n$ of jobs arrives causally, where job J_i arrives at slot/time a_i and must be finished by a deadline of d_i slots starting from a_i or is dropped. We assume $a_i \leq a_{i+1}$ for $1 \leq i < n$. Each job takes one slot to be processed, and if processed before its deadline, job J_i accumulates a payoff/value v_i. A job is *available* at slot t if its absolute deadline is after slot $t-1$, and is *expired* otherwise excluding the already processed jobs. The server can work at variable speed, and can process $k \geq 0$ jobs in any slot by incurring a cost of $g(k)$, where $g(.)$ is a convex function, e.g., $g(k) = k^\alpha, \alpha > 1$.

In a significant departure from prior work on speed scaling, we consider the robust setting, where the online algorithm does not know the deadline for any job, which could potentially be chosen by an adversary. This allows us to model the deadline uncertainty, derive a robust online algorithm, and provide a means to quantify the effect of deadline knowledge on speed scaling. Thus, the information that any online algorithm has at slot t is the set of jobs that have not expired by then, and their respective values. We, however, let the offline optimal algorithm to know the exact deadline and the respective payoff non-causally, to consider the worst case model.

We consider only the deterministic algorithm setting, where on a sequence of jobs $\boldsymbol{\sigma}$, let the set of jobs processed at slot j by an algorithm ALG be P_j. Then the overall profit for ALG is

$$C_{\mathsf{ALG}}(\boldsymbol{\sigma}) = \sum_{j=1}^{\mathsf{last}} (v_{P_j} - g(|P_j|)),$$

where $v_{P_j} = \sum_{i \in P_j} v_j$, and last is the last slot at which Algorithm ALG processes any job. Let $\mathsf{r}_{\mathsf{ALG}}(\boldsymbol{\sigma}) = \frac{C_{\mathsf{OFF}}(\boldsymbol{\sigma})}{C_{\mathsf{ALG}}(\boldsymbol{\sigma})}$, where OFF is the offline optimal algorithm that is allowed to know the sequence $\boldsymbol{\sigma}$ including the job deadlines non-causally. The objective is to find the optimal competitive ratio

$$\mathsf{r}^\star = \min_{\mathsf{ALG}} \max_{\boldsymbol{\sigma}} \mathsf{r}_{\mathsf{ALG}}(\boldsymbol{\sigma}), \tag{1}$$

and the optimal online algorithm achieving $\mathsf{r}^\star$ be OPT. To reiterate, $\min_{\mathsf{ALG}}$ is over all deterministic online algorithms that do not have deadline information of jobs, and make decisions causally at each slot depending on the values of the available jobs only.

▶ **Definition 1.** The effective/incremental cost of processing the k^{th} job in any slot is $c_k = g(k) - g(k-1)$. Since $g(k)$ is convex, $c_k > 0 \ \forall \ k$.

▶ **Definition 2.** For two inputs $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, $\boldsymbol{\sigma}_1 \cup \boldsymbol{\sigma}_2$ corresponds to input where for each slot t, the set of jobs is the union of job arriving at slot t in $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$.

▶ **Lemma 3.** *For any convex function $g(.)$, for any two inputs $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, $C_{\mathsf{OFF}}(\boldsymbol{\sigma}_1 \cup \boldsymbol{\sigma}_2) \leq C_{\mathsf{OFF}}(\boldsymbol{\sigma}_1) + C_{\mathsf{OFF}}(\boldsymbol{\sigma}_2)$.*

Proof. Let the profit (payoff minus the energy cost) accrued in $C_{\mathsf{OFF}}(\boldsymbol{\sigma}_1 \cup \boldsymbol{\sigma}_2)$ corresponding to the processing of jobs belonging to $\boldsymbol{\sigma}_i$ be p_i. From convexity of the cost function $g(.)$, we have $p_i \leq C_{\mathsf{OFF}}(\boldsymbol{\sigma}_i)$ and the result follows since $C_{\mathsf{OFF}}(\boldsymbol{\sigma}_1 \cup \boldsymbol{\sigma}_2) = p_1 + p_2$. ◀

3 min-LCR algorithm

▶ Remark. Since $c_k > 0$ and increasing in k, m (the largest number of jobs that can be processed, where each job has a positive profit) is well defined in min-LCR algorithm.

▶ Remark. A natural Greedy algorithm is a special case of the min-LCR algorithm when $i^\star(\tau) = m$ at all slots τ.

The basic idea behind the min-LCR algorithm has been described in the introduction. To be specific, the online algorithm's profit is $P_{i,\tau}$ if it processes i jobs in slot τ without the knowledge of the deadlines of the available jobs. The online algorithm presumes that possibly no further jobs are going to arrive, and the i-chosen jobs for processing by the algorithm have deadlines as infinity, while the left-over jobs (other than the i chosen jobs) have deadlines that expire in the current slot. The profit of the OFF under this presumption is $o_{i,\tau} = M_{i,\tau} + C_{\mathsf{Greedy}}(i,\tau)$, where $M_{i,\tau}$ is the profit OFF can accrue by processing i highest valued jobs one in each slot starting from slot $\tau + 1$ if their deadlines are infinity, while $C_{\mathsf{Greedy}}(i,\tau)$ is the largest profit possible by processing the set of jobs other than the i highest valued jobs in slot τ itself. The algorithm chooses i that minimizes the ratio of $\frac{o_{i,\tau}}{P_{i,\tau}}$ which essentially is the local competitive ratio at slot τ.

▶ **Theorem 4.** *Algorithm min-LCR is an optimal online algorithm that achieves the optimal competitive ratio* (1).

Algorithm 1: min-LCR algorithm.

Input : Sequence $\boldsymbol{\sigma} = J_1, \ldots, J_n$
At slot τ, consider the union of all the non-processed jobs by the algorithm so far that are available and the newly arriving jobs in slot τ, and call it $E(\tau)$
Arrange the jobs in $E(\tau)$ in non-increasing order of their payoffs/value v
Let $E(\tau)^i = \{\text{first } \ i \text{ jobs in } E(\tau)\}$ and $E^{i-}(\tau) = E(\tau) \backslash E(\tau)^i$
Let $v^{(i)}$ be the value of job of $E(\tau)$ with the i^{th} highest value and $V^{(i)}$ be the sum of the values of the first i jobs with highest values
Let $m = \max_{\{v^{(j)} - [g(j) - g(j-1)] > 0\}} j$
for $i = 1 : m$ **do**
  Let
  $M_{i,\tau} = V^{(i)} - ig(1), P_{i,\tau} = V^{(i)} - g(i), C_{\mathsf{Greedy}}(i,\tau) = \max_{j \in E^{i-}(\tau)} \{V^{(j)} - g(j)\}$
  $\mathsf{LCR}_i(\tau) = \dfrac{M_{i,\tau} + C_{\mathsf{Greedy}}(i,\tau)}{P_{i,\tau}}$
end
Let $i^\star(\tau) = \arg\min_{i=1,\ldots,m} \mathsf{LCR}_i(\tau)$
Process first $i^\star(\tau)$ jobs of $E(\tau)$ at slot τ, call the processed set of jobs $L^p_{\boldsymbol{\sigma}}(\tau)$.

To prove Theorem 4, we show that the competitive ratio of Algorithm min-LCR on input $\boldsymbol{\sigma}$ is at most $\max_\tau \mathsf{LCR}_{i^\star(\tau)}(\tau)$ in Lemma 5, and $\max_\tau \mathsf{LCR}_{i^\star(\tau)}(\tau) \leq \mathsf{r}^\star$ in Lemma 6.

▶ **Lemma 5.** *For any input $\boldsymbol{\sigma}$, $\mathsf{r}_{min-LCR}(\boldsymbol{\sigma}) \leq \max_\tau \mathsf{LCR}_{i^\star(\tau)}(\tau)$.*

Proof. Let $L^p_{\boldsymbol{\sigma}}(\tau)$ be the jobs processed by Algorithm min-LCR on input $\boldsymbol{\sigma}$ at slot τ, and $\bigcup_{\tau=1}^{\mathsf{last}} L^p(\tau) = L^p$, where last is the last slot at which Algorithm min-LCR processes any job.

$$
\begin{aligned}
C_{\mathsf{OFF}}(\boldsymbol{\sigma}) &= C_{\mathsf{OFF}}(L^p \cup \boldsymbol{\sigma} \backslash L^p), \\
&\overset{(a)}{\leq} C_{\mathsf{OFF}}(L^p) + C_{\mathsf{OFF}}(\boldsymbol{\sigma} \backslash L^p) = C_{\mathsf{OFF}}\left(\bigcup_{\tau=1}^{\mathsf{last}} L^p(\tau) \right) + C_{\mathsf{OFF}}(\boldsymbol{\sigma} \backslash L^p), \\
&\overset{(b)}{\leq} \sum_{\tau=1}^{\mathsf{last}} C_{\mathsf{OFF}}(L^p(\tau)) \ + \ C_{\mathsf{OFF}}(\boldsymbol{\sigma} \backslash L^p), \\
&= \sum_{\tau=1}^{\mathsf{last}} C_{\mathsf{OFF}}(L^p(\tau)) \ + \ \sum_{\tau=1}^{\mathsf{last}} C_{\mathsf{OFF}}(\boldsymbol{\sigma} \backslash L^p)_\tau, \qquad (2)
\end{aligned}
$$

where (a) and (b) follow from sub-additivity Lemma (Lemma 3), and where $C_{\mathsf{OFF}}(\boldsymbol{\sigma} \backslash L^p)_\tau$ is the profit obtained by the OFF algorithm in slot τ when the input is only $\boldsymbol{\sigma} \backslash L^p$. Note that this is not true for concave cost functions since sub-additivity lemma (Lemma 3) does not hold. Equation 2 follows from the fact that min-LCR terminates at time slot last leaving behind only unprofitable jobs for slots after last. Hence it is not profitable to process any unprocessed job out of set $\boldsymbol{\sigma} \backslash L^p$ after last.

In order to upper bound $C_{\mathsf{OFF}}(\boldsymbol{\sigma})$ we first observe that the maximum profit that can be made from jobs in set $L^p(\tau)$ is by processing them alone in distinct slots with energy cost of $g(1)$, hence

$$
C_{\mathsf{OFF}}(L^p(\tau)) \leq \sum_{i \in L^p(\tau)} (v_i - g(1)). \qquad (3)
$$

Since $L^p(\tau)$ is the set consisting of first $i^\star(\tau)$ jobs of $E(\tau)$, and $|L^p(\tau)| = i^\star(\tau)$, we get $\sum_{i\in L^p(\tau)} v_i = V^{i^\star(\tau)}$ and $\sum_{i\in L^p(\tau)} g(1) = i^\star(\tau)g(1)$. Hence from (3), we get

$$C_{\mathsf{OFF}}(L^p(\tau)) \le V^{i^\star(\tau)} - i^\star(\tau)g(1) = M_{i^\star,\tau}, \tag{4}$$

where the last inequality follows from the definition of $M_{i^\star,\tau}$ from Algorithm min-LCR. To bound $C_{\mathsf{OFF}}(\boldsymbol{\sigma}\backslash L^p)_\tau$ in (2), we observe that any job among $\boldsymbol{\sigma}\backslash L^p$ that OFF can process at time τ was also available to min-LCR($\boldsymbol{\sigma}$) at time τ but was not processed by min-LCR. Since the maximum profit that can be accrued from all the available packets to min-LCR at time τ is $C_{Greedy}(i^\star,\tau)$ the profit made by OFF($\boldsymbol{\sigma}\backslash L^p$) at time τ must be less than this value.

The set $\boldsymbol{\sigma}\backslash L^p$ consists of all jobs of $\boldsymbol{\sigma}$ that are not processed by the min-LCR algorithm at any slot during its operation. $(\boldsymbol{\sigma}\backslash L^p)_\tau$ is a set of elements of $\boldsymbol{\sigma}\backslash L^p$ which arrived at or before slot τ and whose deadline is after slot $\tau-1$. In comparison, the set $E(\tau)\backslash L^p(\tau)$ is the union of jobs available at slot τ that are never processed by the min-LCR algorithm and the available jobs at slot τ that are processed by the min-LCR algorithm in some slot after slot τ.

Thus,

$$(\boldsymbol{\sigma}\backslash L^p)_\tau \subseteq E(\tau)\backslash L^p(\tau), \tag{5}$$

Similar to the definition of $C_{\mathsf{OFF}}(\boldsymbol{\sigma}\backslash L^p)_\tau$ in (2), let $C_A(\boldsymbol{\sigma}\backslash L^p)_\tau$ be the profit obtained by the algorithm A in slot τ when the input is only $\boldsymbol{\sigma}\backslash L^p$. Then, $C_{\mathsf{OFF}}(\boldsymbol{\sigma}\backslash L^p)_\tau \le \max_{A\in\mathsf{OFF-ALG}} C_A(\boldsymbol{\sigma}\backslash L^p)_\tau$, where the maximization is over all the offline algorithms. Hence, using (5), we get

$$C_{\mathsf{OFF}}(\boldsymbol{\sigma}\backslash L^p)_\tau \le \max_{A\in\mathsf{OFF-ALG}} C_A(E(\tau)\backslash L^p(\tau))_\tau = C_{\mathsf{Greedy}}(i^\star,\tau), \tag{6}$$

where the last equality is derived as follows. Let $A^\star = \arg\max_{A\in\mathsf{OFF-ALG}} C_A(E(\tau)\backslash L^p(\tau))_\tau$ and say it processes some k jobs with values $v_1,\ldots,v_k$ belonging to set $E(\tau)\backslash L^p(\tau)$ in slot τ, then

$$\begin{aligned}\max_{A\in\mathsf{OFF-ALG}} C_A(E(\tau)\backslash L^p(\tau))_\tau &= \sum_{i=1}^{k} v_i \ - g(k),\\ &\overset{(a)}{\le} \sum_{i=1}^{k} v^{(i)} \ - g(k) \ = V^{(k)} - g(k),\\ &\le \max_k\{V^{(k)} - g(k)\} \overset{(b)}{=} C_{\mathrm{Greedy}}(i^\star,\tau),\end{aligned}$$

where (a) follows since sum of the values of any set of k jobs is less than those of the k highest valued jobs (where $v^{(i)}$ is the value of the i^{th} highest valued job), and (b) follows from the definition of $C_{\mathrm{Greedy}}(i^\star,\tau)$ in Algorithm min-LCR since the jobs chosen by $A^\star$ belong to the set $E(\tau)\backslash L^p(\tau)$.

Hence, using (4) and (6), we have from (2),

$$C_{\mathsf{OFF}}(\boldsymbol{\sigma}) \le \sum_{\tau=1}^{\mathsf{last}} M_{i^\star(\tau),\tau} + \sum_{\tau=1}^{\mathsf{last}} C_{\mathsf{Greedy}}(i^\star(\tau),\tau).$$

From the definition of the Algorithm min-LCR, the profit made by it at slot τ is $P_{i^\star(\tau),\tau} = V^{(i^\star(\tau))} - g(i^\star(\tau))$ by processing $i^\star(\tau)$ jobs at slot τ. Thus, the competitive ratio

of min-LCR on input ($\boldsymbol{\sigma}$) algorithm is at most

$$\mathsf{r}_{LCR}(\boldsymbol{\sigma}) = \frac{C_{\mathsf{OFF}}(\boldsymbol{\sigma})}{C_{\mathrm{LCR}}(\boldsymbol{\sigma})} \leq \frac{\sum_{\tau=1}^{\mathsf{last}}(M_{i^\star(\tau),\tau} + C_{\mathsf{Greedy}}(i^\star(\tau),\tau))}{\sum_{\tau=1}^{\mathsf{last}} P_{i^\star(\tau),\tau}} \leq \max_{\tau} \frac{M_{i^\star(\tau),\tau} + C_{\mathsf{Greedy}}(i^\star(\tau),\tau)}{P_{i^\star(\tau),\tau}},$$
$$= \max_{\tau} \mathsf{LCR}_{i^\star(\tau)}(\tau). \tag{7}$$

◀

3.1 Lower Bound

Next, to complete the proof of Theorem 4, we show that the optimal competitive ratio $\mathsf{r}^\star$ is lower bounded by $\max_\tau \mathsf{LCR}_{i^\star(\tau)}(\tau)$ for any input $\boldsymbol{\sigma}$ and any slot τ. The main idea in the proof is that if there exists an optimal online algorithm OPT which has a strictly better competitive ratio ($\mathsf{r}^\star$) than min-LCR, then there must exist an input ($\boldsymbol{\sigma}_1$) at which $\frac{\mathsf{OFF}(\boldsymbol{\sigma}_1)}{LCR(\boldsymbol{\sigma}_1)} > \mathsf{r}^\star$ and $\frac{\mathsf{OFF}(\boldsymbol{\sigma}_1)}{\mathsf{OPT}(\boldsymbol{\sigma}_1)} \leq \mathsf{r}^\star$. Using this input we then construct another input ($\boldsymbol{\sigma}_2$) for which $\frac{\mathsf{OFF}(\boldsymbol{\sigma}_2)}{\mathsf{OPT}(\boldsymbol{\sigma}_2)} > \mathsf{r}^\star$ contradicting the assumption that competitive ratio of OPT is $\mathsf{r}^\star$.

▶ **Lemma 6.** *$\mathsf{r}^\star \geq \max_\tau \mathsf{LCR}_{i^\star(\tau)}(\tau)$ for any input $\boldsymbol{\sigma}$ and time slot τ.*

Contradiction. Let the hypothesis H_1 : $\exists\ \tau, \boldsymbol{\sigma}_1$ such that $\mathsf{LCR}_{i^\star(\tau)}(\tau) > \mathsf{r}^\star$. For ease of exposition in this proof, let $\mathsf{LCR}^{(\tau)}_{\boldsymbol{\sigma}_1} = \mathsf{LCR}_{i^\star(\tau)}(\tau)$. Consider the set of jobs $E(\tau)$ at slot τ that is the union of all the non-processed and non-expired jobs till slot $\tau - 1$ by the Algorithm min-LCR and the newly arrived jobs at slot τ. Let $v_1, \ldots, v_{|E(\tau)|}$ be the values of these jobs, where job i arrived at slot a_i and has deadline d_i.

We construct another input sequence $\boldsymbol{\sigma}_2$ that consists of $|E(\tau)|$ jobs with values $v_1, \ldots, v_{|E(\tau)|}$. All the jobs of $\boldsymbol{\sigma}_2$ arrive at slot 1, and the deadlines for each job is equal to the $d_i - \tau - a_i$. Important to note that d_i is allowed to be arbitrary and unknown to the online algorithm while known to the OFF in both $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$.

Consider a new hypothesis H_2 : Optimum online algorithm OPT on input $\boldsymbol{\sigma}_2$ has competitive ratio lower than $\mathsf{LCR}^{(1)}_{\boldsymbol{\sigma}_2}$. It is easy to check that from the min-LCR algorithm definition, that $\mathsf{LCR}^{(1)}_{\boldsymbol{\sigma}_2} = \mathsf{LCR}^{(\tau)}_{\boldsymbol{\sigma}_1}$. From hypothesis H_1, $\mathsf{LCR}^{(\tau)}_{\boldsymbol{\sigma}_1} > \mathsf{r}^\star$, which implies that there exists an optimum online algorithm OPT that on input $\boldsymbol{\sigma}_2$ has competitive ratio lower than $\mathsf{LCR}^{(1)}_{\boldsymbol{\sigma}_2}$, since $\mathsf{r}^\star$ is achievable. Hence $H_1 \implies H_2$. Now we will contradict hypothesis H_2.

Let the optimal online algorithm OPT on input $\boldsymbol{\sigma}_2$ process any $1 \leq k \leq |E(\tau)|$ jobs at slot 1, leaving the remaining jobs for later slots. Since the deadlines d_i are arbitrary, let the true deadlines for the k jobs (whatever the choice of k may be for OPT) that were processed by OPT in slot 1 to be ∞, while keeping the deadlines for all jobs other than k selected ones to be slot 1 itself. Thus, there are no available jobs at slot 2 for OPT. Thus, the OPT can make the maximum profit by sending the k highest valued jobs of $\boldsymbol{\sigma}_2$ or $E(\tau)$ in slot 1.

Given that offline optimal OFF knows the deadlines, OFF processes the k jobs chosen by OPT for processing in slot 1 in k slots starting from the second slot individually, and among the rest of $|E(\tau)| - k$ jobs processes as many jobs in slot 1 to maximize its profit in slot 1, which by definition is $C_{\mathsf{Greedy}}(k, 1)$. Thus, $C_{\mathsf{OFF}}(\boldsymbol{\sigma}_2) = M_{k,1} + C_{\mathsf{Greedy}}(k, 1)$, while $C_{\mathsf{OPT}}(\boldsymbol{\sigma}_2) = \sum_{j=1}^{j=k} v^{(j)} \ - g(k) = P_{k,1}$. Therefore,

$$\frac{C_{\mathsf{OFF}}(\boldsymbol{\sigma}_2)}{C_{\mathsf{OPT}}(\boldsymbol{\sigma}_2)} = \frac{M_{k,1} + C_{\mathsf{Greedy}}(k,1)}{P_{k,1}}, = \mathsf{LCR}_k(1),$$
$$\geq \mathsf{LCR}_{i^\star(1)}(1).$$

since $\mathsf{LCR}_{i^\star(1)}(1)$ is the minimum LCR over all possible k. Thus, the optimum online algorithm OPT on input $\boldsymbol{\sigma}_2$ cannot have a competitive ratio strictly lower than $\mathsf{LCR}_{i^\star(1)}(1) = \mathsf{LCR}^{(1)}_{\boldsymbol{\sigma}_2}$. Thus, we get contradiction to hypothesis H_2 and equivalently to H_1. ◀

▶ Remark. The proof idea presented above is always an appealing candidate to show the optimality of any given online algorithm, however, fails most often. The key insight that makes it work for this problem is that the criteria used (minimize the LCR) by the min-LCR algorithm to select the number of packets to process at any time is an also an upper bound on the competitive ratio of the min-LCR algorithm. The typical methods to show lower bounds for online algorithms are: either by explicit construction of a lower bound example, or using Yao's min-max Lemma for randomized algorithms. The novelty with our approach is that the technique surprisingly works for any general convex energy cost function, and without having to explicitly evaluate the lower or upper bounds, which is very rarely found in literature.

After establishing that the min-LCR algorithm is an optimal online algorithm for any convex cost function g, we next concentrate on a specific cost function $g(k) = k^\alpha$ for $\alpha \geq 2$ that is the most popular choice in literature, to derive some concrete bounds on the competitive ratio of the min-LCR algorithm. Recall that min-LCR algorithm involves finding the number of jobs k that minimizes the LCR_k, which can be exhaustive. We next propose a simplified version of the min-LCR algorithm where exactly $k = \lfloor \beta m \rfloor$ or $k = \lceil \beta m \rceil$ jobs are processed in each slot depending on whichever one has lower LCR, and where β is the solution of the equation $x^\alpha + x^{\alpha-1} - 1 = 0$. We also consider the natural greedy special case of min-LCR algorithm that always processes m jobs, where m has been defined in the min-LCR algorithm.

We first derive a lower bound on the competitive ratio of any online algorithm, and next show that the simplified min-LCR algorithm (and consequently the min-LCR algorithm) achieves that lower bound for $\alpha = 2$. For $\alpha > 2$, we show that the competitive ratio of the simplified min-LCR algorithm is at most 3, while for $\alpha \geq 2.5$ it is at most 2.618.

4 $g(k) = k^\alpha$ for $\alpha \geq 2$

4.1 Lower Bound on the Competitive Ratio for $\alpha = 2$

▶ **Lemma 7.** *For any online algorithm* ALG, $\mathsf{r}_{\mathsf{ALG}} \geq \phi + 1$ *for cost function* $g(k) = k^\alpha$ *with* $\alpha = 2$, *where* $\phi = 1/\delta$ *and* $\delta = \frac{\sqrt{5}-1}{2}$.

Proof. We consider $2z$ jobs each with value $2z$ arriving at the start of slot 1. Thus, in slot 1 at most z jobs will be processed[1] by any algorithm since the cost function is $g(k) = k^2$, and the effective cost of processing job $z+1$ or higher is at least $g(z+1) - g(z) > 2z$. Let an online algorithm ALG choose to process k jobs out of the maximum possible z. Then we adversarially choose the deadlines of these k jobs to be infinity ∞, while keeping the deadlines of all other remaining jobs to be slot 1 itself. Since the offline optimal OFF knows the deadlines, it processes the maximum possible m jobs in slot 1, while processes the k jobs chosen by the ALG for slot 1, one at a time starting from slot 2 in k slots, making a profit of $2z \cdot z - z^2 + 2z \cdot k - k$, while the ALG makes only a profit of $2zk - k^2$. Thus, the competitive ratio of any online algorithm ALG as a function of k is

$$\mathsf{r}_{\mathsf{ALG}} \geq \min_k \frac{2z \cdot z - z^2 + 2z \cdot k - k}{2zk - k^2}.$$

[1] Processing any more jobs is unprofitable.

We take the limit as $z \to \infty$ to get that

$$\mathsf{r}_{\mathsf{ALG}} \geq \lim_{z\to\infty} \min_{k} \frac{2z \cdot z - z^2 + 2z \cdot k - k}{2zk - k^2} = \lim_{z\to\infty} \min_{k} \frac{2z \cdot z - z^2 + 2z \cdot k}{2zk - k^2},$$

since $\lim_{z\to\infty} \frac{-k}{2zk-k^2} = 0$ for any choice of k including the optimizer. It is easy to show that for $\delta = \frac{\sqrt{5}-1}{2}$, (see Appendix A.3)

$$\delta z = \arg\min_{k} \frac{2z \cdot z - z^2 + 2z \cdot k}{2zk - k^2}, \tag{8}$$

and $\lim_{m\to\infty} \frac{2z\cdot z - z^2 + 2z\cdot k}{2zk-k^2}|_{k=\frac{\sqrt{5}-1}{2}z} = \phi + 1$, where $\phi = \frac{1}{\delta}$. ◀

One can proceed similarly and get an expression for the lower bound on the competitive ratio for all values of $\alpha > 2$ as

$$\max_{z\in\mathbb{Z}^+} \max_{x\leq g(z+1)-2g(z)+g(z-1)} \min_{1\leq k\leq z} \left\{ \frac{[k(z^\alpha - (z-1)^\alpha + x - 1)] + [z(z^\alpha - (z-1)^\alpha + x) - z^\alpha]}{[k(z^\alpha - (z-1)^\alpha + x) - k^\alpha]} \right\}, \tag{9}$$

however, it is not easy to simplify it analytically, needing a numerical solution as presented in Fig. 1. To be more concrete, we next derive a slightly loose lower bound for $\alpha > 2$ as follows.

4.2 Lower Bound on the Competitive Ratio for $\alpha > 2$

▶ **Lemma 8.** *For any online algorithm* ALG, $\mathsf{r}_{\mathsf{ALG}} \geq \sqrt{2} + 1$ *for* $g(k) = k^\alpha$ *for all* $\alpha > 2$.

Proof. Consider the input where four jobs arrive at the beginning of slot 1 each with value $v = \left(1 + \frac{1}{\sqrt{2}}\right)[g(2) - \sqrt{2}g(1)]$. Any online algorithm ALG can process $k = 1$ or 2 or 3 or 4 jobs in slot 1. The choice of v is such that $v < g(3) - g(2)$, where $g(3) - g(2)$ is the effective cost of processing the third job in slot 1. Moreover, since g is a convex function $g(4) - g(3) > g(3) - g(2)$. Therefore processing the third or the fourth job in slot 1 incurs negative profit, and hence at most 2 jobs will be processed in slot 1.

Now depending on the choice of ALG for $k = 1, 2$, the adversarial choice of deadline will be that those k jobs will have deadline ∞, while the remaining $4 - k$ jobs will have deadline as slot 1 itself. A simple enumerative exercise reveals that $\mathsf{r}_{\mathsf{ALG}}|_{k=1} = \sqrt{2}+1, \mathsf{r}_{\mathsf{ALG}}|_{k=2} = \sqrt{2}+1$. The choice of v is the only non-trivial part in deriving this lower bound. ◀

4.3 Upper Bound on the Competitive Ratio for $\alpha \geq 2$

From here onwards we derive upper bounds on the competitive ratio of a simplified min-LCR algorithm, where we choose a particular number of jobs to process in every slot. The simplified min-LCR algorithm is as follows.

▶ **Lemma 9.** *The competitive ratio of sim-LCR Algorithm is at most* $\phi + 1$ *for* $\alpha = 2$ *or* $\alpha \geq 2.5$.

Combining Lemma 7 and Lemma 9, we get the following result.

▶ **Corollary 10.** *sim-LCR Algorithm is an optimal online algorithm for* $\alpha = 2$ *and the optimal competitive ratio for* $\alpha = 2$ *is* $\phi + 1$.

Algorithm 2: Simplified min-LCR algorithm (sim-LCR)

Input : Sequence $\boldsymbol{\sigma} = J_1, \ldots, J_n$

At slot τ, consider the union of all the non-processed jobs by the algorithm so far that have not-expired and the newly arriving jobs in slot τ and call it $E(\tau)$

Arrange the jobs in $E(\tau)$ in non-increasing order of their payoffs/value v

Let $E(\tau)^i = \{\text{first } i \text{ jobs in } E(\tau)\}$ and $E^{i-}(\tau) = E(\tau) \backslash E(\tau)^i$

Let $v^{(i)}$ be the value of job of $E(\tau)$ with the i^{th} highest value, $V^{(i)}$ be the sum of the values of the first i jobs with highest values

Let $m = \max_{\{v^{(j)} - [g(j) - g(j-1)] > 0\}} j$

Let $M_{i,\tau} = V^{(i)} - ig(1), P_{i,\tau} = V^{(i)} - g(i), C_{\mathsf{Greedy}}(i,\tau) = \max_{j \in E^{i-}(\tau)} \{V^{(j)} - g(j)\}$

Let β be the solution of $x^\alpha + x^{\alpha-1} - 1 = 0$

Compute $\mathsf{LCR}_i(\tau) = \dfrac{M_{i,\tau} + C_{\mathsf{Greedy}}(i,\tau)}{P_{i,\tau}}$ as in min-LCR algorithm for $i = \lfloor \beta m \rfloor$ or $i = \lceil \beta m \rceil$

Process either $i = \lfloor \beta m \rfloor$ or $i = \lceil \beta m \rceil$ jobs whichever one has lower LCR_i in slot τ.

Proof of Lemma 9. Let $i(\tau)$ be the optimizer among $i = \lfloor \beta m \rfloor$ or $i = \lceil \beta m \rceil$ that minimizes the LCR_i with the sim-LCR algorithm at slot τ. Then following an identical proof as for Lemma 5, it follows that the competitive ratio of the sim-LCR algorithm from (7),

$$\mathsf{r}_{\text{sim-LCR}}(\boldsymbol{\sigma}) = \frac{C_{\mathsf{OFF}}(\boldsymbol{\sigma})}{C_{\text{sim-LCR}}(\boldsymbol{\sigma})} \leq \frac{\sum_{\tau=1}^{\mathsf{last}} M_{i(\tau),\tau} + C_{\mathsf{Greedy}}(i(\tau),\tau)}{\sum_{\tau=1}^{\mathsf{last}} P_{i(\tau),\tau}} \leq \max_{\tau} \frac{M_{i(\tau),\tau} + C_{\mathsf{Greedy}}(i(\tau),\tau)}{P_{i(\tau),\tau}} = \max_{\tau} \mathsf{LCR}_{i(\tau)}(\tau). \tag{10}$$

Thus, to complete the proof, in the following, we show that the LCR_k for either $k = \lfloor \beta m \rfloor$ or $k = \lceil \beta m \rceil$ is less than $\phi + 1$ for the sim-LCR Algorithm at all slots τ. From the definition of the LCR

$$\mathsf{LCR}_k = \frac{M_{k,\tau} + C_{\mathsf{Greedy}}(k,\tau)}{P_{k,\tau}} \overset{(a)}{=\!\leq} \frac{\sum_{j=1}^{j=k} v^{(j)} \; - \; k \cdot g(1) + \sum_{j=1}^{j=m} v^{(j)} \; - \; g(m)}{\sum_{j=1}^{j=k} \; v^{(j)} \; - \; g(k)}, \tag{11}$$

(a) follows from the definition of $M_{k,\tau}$ and an upper bound derived for $C_{\mathsf{Greedy}}(k,\tau)$ in Appendix A.2. Recall that the set of available jobs $E(\tau)$ at each slot τ are arranged in decreasing order of their values, hence, for $k \leq m$, the average of the values of the first k jobs is greater than the average of the values of the first m. Hence, we have $\frac{m}{k} \sum_{j=1}^{j=k} v^{(j)} \geq \sum_{j=1}^{j=m} v^{(j)}$. Substituting this in (11) yields

$$\begin{aligned} \mathsf{LCR}_k &\leq \frac{\sum_{j=1}^{j=k} v^{(j)} \; - \; k + \frac{m}{k}\left(\sum_{j=1}^{j=k} v^{(j)}\right) \; - \; g(m)}{\sum_{j=1}^{j=k} \; v^{(j)} \; - \; g(k)} \\ &= \frac{m}{k} + 1 + \frac{g(k) + \frac{m}{k} g(k) - g(m) - k}{\sum_{j=1}^{j=k} \; v^{(j)} \; - \; g(k)}. \end{aligned} \tag{12}$$

Consider the numerator of the second term $f(k) = g(k) + \frac{m}{k} g(k) - g(m) - k$ at $k = \beta m$. By definition, $f(\beta m) = m^\alpha(\beta^\alpha + \beta^{\alpha-1} - 1) - \beta m \leq 0$ since β is the solution of the equation $x^\alpha + x^{\alpha-1} - 1 = 0$. Moreover, $f(\lfloor \beta m \rfloor) = (\lfloor \beta m \rfloor)^\alpha + \frac{m}{(\lfloor \beta m \rfloor)} (\lfloor \beta m \rfloor)^\alpha - m^\alpha - \lfloor \beta m \rfloor \leq 0$, since $(\lfloor \beta m \rfloor)^\alpha + \frac{m}{(\lfloor \beta m \rfloor)} (\lfloor \beta m \rfloor)^\alpha - m^\alpha \leq m^\alpha(\beta^\alpha + \beta^{\alpha-1} - 1)$. Using this in (12), we get

$$\mathsf{LCR}_{\lfloor \beta m \rfloor} \leq \frac{m}{k} + 1. \tag{13}$$

For $\alpha \geq 2.5$, by direct computation, one can check that for $m = \{1, 3, 6, 8, 9, 10, 11, 12\}$ and $\forall m \geq 13, \mathsf{LCR}_{\lfloor \beta m \rfloor} \leq \phi + 1$. For the remaining values of $m = 2, 4, 5, 7$ either $\mathsf{LCR}_{\lfloor \beta m \rfloor} \leq \phi + 1$ or $\mathsf{LCR}_{\lceil \beta m \rceil} \leq \phi + 1$, as derived in Appendix A.4.

For $\alpha = 2$, a little more involved approach is needed as follows. For $\alpha = 2$, $\beta = \delta$, where β is the solution of the equation $x^\alpha + x^{\alpha-1} - 1 = 0$. Let γ be the positive root of the equation $x^2 + (m-1)x - m^2 = 0$. From (11), using $\alpha = 2$,

$$\mathsf{LCR}_k = \frac{m}{k} + 1 + \frac{k^2 + mk - m^2 - k}{\sum_{j=1}^{j=k} v^{(j)} - k^2}. \tag{14}$$

The quadratic equation $k^2 + mk - m^2 - k$ is increasing in the interval $[\delta m, \gamma]$ ($\delta = \frac{\sqrt{5}-1}{2}$) with value < 0 at $k = \delta m$ and $= 0$ at $k = \gamma$ (by definition of γ). So at all intermediate values of $k \in (\delta m, \gamma)$, the value of the equation must be less than 0. Thus, if $\lceil \delta m \rceil \in [\delta m, \gamma]$, we have $\mathsf{LCR}_{\lceil \delta m \rceil} \leq \frac{m}{\lceil \delta m \rceil} + 1 \leq \frac{m}{\delta m} + 1 = \phi + 1$.

Next, we consider the case when $\lceil \delta m \rceil \in (\gamma, \delta m + 1]$. The numerator of the third term in the RHS of (14) is an increasing function for $(\gamma, \delta m + 1]$ from the definition of γ. Thus, since $\sum_{j=1}^{j=k} v^{(j)} \geq k[g(m) - g(m-1)]$ where $v^{(j)}$ is the j^{th} highest value of the job when arranged in a non-increasing order of values. Thus, from (14), we get

$$\mathsf{LCR}_k \leq \frac{m}{k} + 1 + \frac{k^2 + mk - m^2 - k}{(2m-1)k - k^2}. \tag{15}$$

Let $\psi(k) = \frac{m}{k} + 1 + \frac{k^2 + mk - m^2 - k}{(2m-1)k - k^2}$, then it follows that $\frac{\partial \psi(k)}{\partial k} \geq 0$ for $k \in (\gamma, \delta m + 1]$, and $\psi(\delta m + 1) \leq \phi + 1$ for $m \geq 5$. This implies that $\mathsf{LCR}_k \leq \phi + 1$ for all $k \in (\gamma, \delta m + 1]$. Thus, even if $\lceil \delta m \rceil \in (\gamma, \delta m + 1]$, $\mathsf{LCR}_{\lceil \beta m \rceil} \leq \phi + 1$ for $m \geq 5$. For $m = 1$, $\mathsf{LCR}_{\lceil \delta m \rceil} \leq 2$, while for

$$m = 3 \implies LCR_{\lceil \delta m \rceil} \leq \frac{3}{2} + 1 + \frac{2^2 + 3 \cdot 2 - 3^2 - 2}{(2 \cdot 3 - 1)2 - 2^2} < \phi + 1 \tag{16}$$

$$m = 4 \implies LCR_{\lceil \delta m \rceil} \leq \frac{4}{3} + 1 + \frac{3^2 + 4 \cdot 3 - 4^2 - 3}{(2 \cdot 4 - 1)3 - 3^2} < \phi + 1. \tag{17}$$

For $m = 2$, the proof is identical to the one for the case of $\alpha \geq 2.5$ for $m = 2$ as provided in Appendix A.4. ◀

4.4 Upper Bound on the Competitive Ratio of the min-LCR algorithm for $\alpha \geq 2$

We can obtain a slightly loose upper bound for all values of $\alpha \geq 2$ by enforcing the min-LCR algorithm to process all m jobs, where m is largest number of jobs that can be processed at any slot with positive profit for each job. This restriction can essentially be seen as a greedy analogue of the min-LCR algorithm, and we call it Greedy.

▶ **Lemma 11.** *The competitive ratio of the* Greedy *algorithm (min-LCR algorithm with $i^\star = m$) is at most* 3 *for all $\alpha > 2$.*

Proof can be found in Appendix A.5.

References

1 Susanne Albers, Fabian Müller, and Swen Schmelzer. Speed scaling on parallel processors. *Algorithmica*, 68(2):404–425, 2014.

2 Yossi Azar, Nikhil R Devanur, Zhiyi Huang, and Debmalya Panigrahi. Speed scaling in the non-clairvoyant model. In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*, pages 133–142. ACM, 2015.

3 Yossi Azar, Bala Kalyanasundaram, Serge Plotkin, Kirk R Pruhs, and Orli Waarts. Online load balancing of temporary tasks. In *Workshop on algorithms and data structures*, pages 119–130. Springer, 1993.

4 Nikhil Bansal, Ho-Leung Chan, Tak-Wah Lam, and Lap-Kei Lee. Scheduling for speed bounded processors. In *International Colloquium on Automata, Languages, and Programming*, pages 409–420. Springer, 2008.

5 Nikhil Bansal, Ho-Leung Chan, Kirk Pruhs, and Dmitriy Katz. Improved bounds for speed scaling in devices obeying the cube-root rule. *Automata, languages and programming*, pages 144–155, 2009.

6 Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM (JACM)*, 54(1):3, 2007.

7 Nikhil Bansal, Kirk Pruhs, and Cliff Stein. Speed scaling for weighted flow time. *SIAM Journal on Computing*, 39(4):1294–1308, 2009.

8 Neal Barcelo, Peter Kling, Michael Nugent, and Kirk Pruhs. *Optimal Speed Scaling with a Solar Cell*, pages 521–535. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-48749-6_38.

9 Sanjoy Baruah, Gilad Koren, Bhubaneswar Mishra, Arvind Raghunathan, Louis Rosier, and Dennis Shasha. On-line scheduling in the presence of overload. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 100–110. IEEE, 1991.

10 Ho-Leung Chan, Joseph Wun-Tat Chan, Tak-Wah Lam, Lap-Kei Lee, Kin-Sum Mak, and Prudence WH Wong. Optimizing throughput and energy in online deadline scheduling. *ACM Transactions on Algorithms (TALG)*, 6(1):10, 2009.

11 Ho-Leung Chan, Jeff Edmonds, Tak-Wah Lam, Lap-Kei Lee, Alberto Marchetti-Spaccamela, and Kirk Pruhs. Nonclairvoyant speed scaling for flow and energy. *Algorithmica*, 61(3):507–517, 2011.

12 Aaron Coté, Adam Meyerson, Alan Roytman, Michael Shindler, and Brian Tagiku. Energy-efficient online scheduling with deadlines. *unpublished manuscript*, 2010.

13 Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Nonclairvoyantly scheduling power-heterogeneous processors. *Sustainable Computing: Informatics and Systems*, 1(3):248–255, 2011.

14 Xin Han, Tak-Wah Lam, Lap-Kei Lee, Isaac KK To, and Prudence WH Wong. Deadline scheduling and power management for speed bounded processors. *Theoretical Computer Science*, 411(40-42):3587–3600, 2010.

15 Sandy Irani, Sandeep Shukla, and Rajesh Gupta. Algorithms for power savings. *ACM Transactions on Algorithms (TALG)*, 3(4):41, 2007.

16 Tak-Wah Lam, Lap-Kei Lee, Isaac KK To, and Prudence WH Wong. Speed scaling functions for flow time scheduling based on active job count. In *European Symposium on Algorithms*, pages 647–659. Springer, 2008.

17 Marco Riedel. Online request server matching. *Theoretical computer science*, 268(1):145–160, 2001.

18 Adam Wierman, Lachlan LH Andrew, and Ao Tang. Power-aware speed scaling in processor sharing systems. In *INFOCOM 2009, IEEE*, pages 2007–2015. IEEE, 2009.

19 Frances Yao, Alan Demers, and Scott Shenker. A scheduling model for reduced cpu energy. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 374–382. IEEE, 1995.

A Appendix

A.1 Numerical Evaluation of the Lower bound (9)

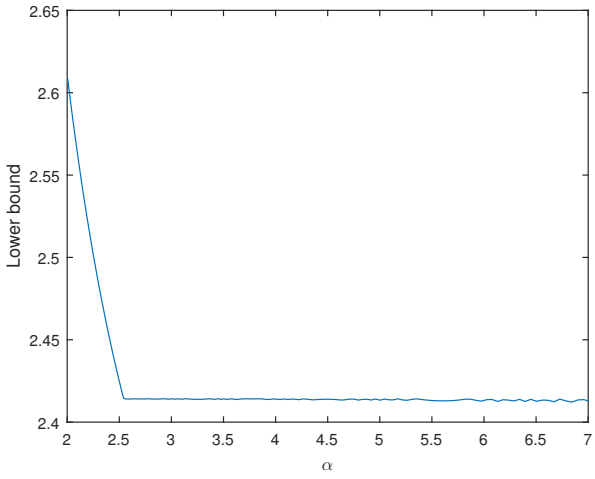

Figure 1 Numerical Evaluation of the Lower bound (9) as a function of α.

A.2 Proof for upper bounding $C_{\mathsf{Greedy}}(k,\tau)$ in (11)

▶ **Lemma 12.**

$$C_{\mathsf{Greedy}}(k,\tau) \leq \sum_{j=1}^{j=m} v^{(j)} - g(m).$$

Proof. Recall that the effective cost of the i^{th} job is $c_i = g(i) - g(i-1)$.

$$\begin{aligned} C_{\mathsf{Greedy}}(k,\tau) &\overset{(a)}{=} \max_{j \in E^{k-}(\tau)} (V^{(j)} - g(j)), \\ &\overset{(b)}{\leq} \max_{j \in E(\tau)} (V^{(j)} - g(j)), \\ &\overset{(c)}{=} (V^{(j^\star)} - g(j^\star)) = \sum_{i=1}^{j^\star} v^{(i)} - c_i, \\ &\overset{(d)}{=} \sum_{j=1}^{j=m} v^{(j)} - g(m), \end{aligned}$$

(a) follows from the definition, (b) follows since $E^{k-}(\tau) \subseteq E(\tau)$, (c) follows by defining the optimizer j in (b) to be $j^\star$, while (d) is true because of the following inequalities

$$\forall i > m: \quad v^{(i)} \overset{c}{\leq} v^{(m+1)} \overset{d}{\leq} c_{m+1} \overset{e}{\leq} c_i \implies [v^{(i)} - c_i] \leq 0$$

$$\forall i \le m: \quad v^{(i)} \overset{c}{\ge} v^{(m)} \overset{d}{\ge} c_m \overset{e}{\ge} c_i \implies [v^{(i)} - c_i] \ge 0,$$

where (c) follows since jobs are indexed in the non-increasing order of their values, and (d) follows from the definition of m, and (e) follows from the convexity of $g(k)$. ◀

A.3 Proof of (8)

▶ **Lemma 13.**

$$\arg\min_k \left\{ \frac{z^2 + 2zk}{2zk - k^2} \right\} = \delta z.$$

Proof. Taking the derivative, we get

$$\frac{\partial}{\partial k}\left(\frac{2zk + z^2}{2zk - k^2}\right) = \frac{2z(k^2 + zk - z^2)}{(2zk - k^2)^2}, \tag{18}$$

which when equated to zero, we get $k^\star = \delta z$. Taking the second derivative and evaluating at δm, we show that $k^\star = \delta m$ is a local minima as follows.

$$\frac{\partial^2}{\partial k^2}\left(\frac{2zk + z^2}{2zk - k^2}\right) = \frac{[2z(2zk - k^2)]\,[2z^3 - 6z^2k + 4z^3 + 3zk^2]}{(2zk - k^2)^4},$$

$$\implies \frac{\partial^2}{\partial k^2}\left(\frac{2zk + z^2}{2zk - k^2}\right)\Bigg|_{k=\delta z} \overset{(a)}{>} 0,$$

where (a) follows since

$$[2k^3 - 6z^2k + 4z^3 + 3zk^2]\Big|_{k=\delta z} = [2z^3(2\delta^3 + 3\delta^2 - 6\delta + 4)] > 0.$$

Evaluating at the boundary points of $k = 1$, we get $\frac{z^2 + 2z}{2z - 1} > \phi + 1$ and at $k = z$, $\frac{z^2 + 2z^2}{2z^2 - z^2} = 3 \ge \phi + 1$. Thus, we conclude that $k = \delta z$ is the minima. ◀

A.4 Proof of bounding $\mathsf{LCR}_{\lfloor \beta m \rfloor}$ and $\mathsf{LCR}_{\lceil \beta m \rceil}$ for $m = 2, 4, 5, 7$

▶ **Lemma 14.** *For $m = \{2, 4, 5, 7\}$ $\forall \alpha \ge 2.5$ either $\mathsf{LCR}_{\lfloor \beta m \rfloor}$ or $\mathsf{LCR}_{\lceil \beta m \rceil}$ is $\le \phi + 1$.*

Proof. $m = 2$: $\alpha \ge 2 \implies \lfloor \beta m \rfloor = 1$ and $\lceil \beta m \rceil = 2$. When $v^{(2)} \le \left(1 + \frac{1}{\sqrt{2}}\right)[g(2) - \sqrt{2}g(1)]$,

$$\mathsf{LCR}_1 \overset{p}{=} \frac{[v^{(1)} - g(1)] + [v^{(2)} + v^{(3)} - g(2)]}{[v^{(1)} - g(1)]} \overset{q}{\le} 1 + \frac{2v^{(2)} - g(2)}{v^{(2)} - g(1)} \overset{r}{\le} \phi + 1,$$

while when $v^{(2)} \ge \left(1 + \frac{1}{\sqrt{2}}\right)[g(2) - \sqrt{2}g(1)]$,

$$\mathsf{LCR}_2 \overset{p}{=} \frac{[v^{(1)} + v^{(2)} - 2g(1)] + [v^{(3)} + v^{(4)} - g(2)]}{[v^{(1)} + v^{(2)} - g(2)]} \overset{q}{\le} 1 + \frac{2v^{(2)} - 2g(1)}{2v^{(2)} - g(2)} \overset{r}{\le} \phi + 1,$$

where p follows from the definition of LCR, q follows from the fact that $v^{(1)} \ge v^{(2)} \ge v^{(3)} \ge v^{(4)}$ and r follows by simple evaluation in the appropriate range of choice for $v^{(2)}$. $m = 4$: When $\alpha \in [2.5, 2.945] \implies \lceil \beta m \rceil = \lceil \beta * 4 \rceil = 3$

$$\implies \mathsf{LCR}_{\lceil \beta m \rceil} \overset{a}{=} \frac{4}{3} + 1 + \frac{7 \cdot 3^{(\alpha-1)} - 4 \cdot 4^{(\alpha-1)}}{12(4^{(\alpha-1)} - 3^{(\alpha-1)})} \overset{b}{\le} 2 + \frac{1}{4\left(\frac{4^{(\alpha-1)}}{3^{(\alpha-1)}} - 1\right)} \overset{c}{\le} \phi + 1,$$

$$\alpha \geq 2.945 \implies \lfloor \beta m \rfloor \geq 3 \implies \mathsf{LCR}_{\lfloor \beta m \rfloor} \overset{d}{\leq} \frac{m}{k} + 1 = \frac{4}{3} + 1 \leq \phi + 1.$$

$m = 5$: When $\alpha \in [2.5, 3.641] \implies \lceil \beta m \rceil = 4$

$$\implies \mathsf{LCR}_{\lceil \beta m \rceil} \overset{a}{\leq} \frac{5}{4} + 1 + \frac{9 \cdot 4^{(\alpha-1)} - 5 \cdot 5^{(\alpha-1)}}{20(5^{(\alpha-1)} - 4^{(\alpha-1)})} \overset{b}{=} 2 + \frac{1}{5\left(\frac{5^{(\alpha-1)}}{4^{(\alpha-1)}} - 1\right)} \overset{e}{\leq} \phi + 1,$$

$$\alpha \geq 3.641 \implies \lfloor \beta m \rfloor \geq 4 \implies \mathsf{LCR}_{\lfloor \beta m \rfloor} \overset{d}{\leq} \frac{m}{k} + 1 = \frac{5}{4} + 1 \leq \phi + 1.$$

$m = 7$:

$$\alpha \in [2.5, 2.6] \implies \lceil \beta m \rceil = 5 \implies \mathsf{LCR}_{\lceil \beta m \rceil} \overset{a}{\leq} \frac{7}{5} + 1 + \frac{\frac{12}{5}5^{\alpha} - 7^{\alpha} - 5}{5[7^{\alpha} - 6^{\alpha} - 5^{\alpha}]} \leq \phi + 1,$$

$$\alpha \geq 2.6 \implies \lfloor \beta m \rfloor \geq 5 \implies \mathsf{LCR}_{\lfloor \beta m \rfloor} \overset{d}{\leq} \frac{m}{k} + 1 = \frac{7}{5} + 1 \leq \phi + 1.$$

In all the cases, a follows from (12) along with the fact that $\sum_{j=1}^{j=k} v^{(j)} \geq k[g(m) - g(m-1)]$ and $g(k) = k^{\alpha}$, (b) follows by re-arranging terms, c is true because $\forall\ \alpha \geq 2.5$ we have $\left(\frac{4^{(\alpha-1)}}{3^{(\alpha-1)}} - 1\right) - 1 \geq 0.53$, d follows from (13), and e is true because $\forall \alpha \geq 2.5$ we have $\left(\frac{5^{(\alpha-1)}}{4^{(\alpha-1)}} - 1\right) \geq 0.39$. ◀

A.5 Proof of Lemma 11

Proof. From (10), $\mathsf{r}_{\text{sim-LCR}}(\boldsymbol{\sigma}) \leq \max_{\tau} \mathsf{LCR}_{i(\tau)}$. In this greedy case, $i(\tau) = m$, always, and hence to prove the result we show that $\mathsf{LCR}_m \leq 3$ for all slot τ and all inputs $\boldsymbol{\sigma}$. From (11), we have that $\mathsf{LCR}_k \leq \frac{m}{k} + 1 + \frac{g(k) + \frac{m}{k}g(k) - g(m) - k}{\sum_{j=1}^{j=k} v^{(j)} - g(k)}$. Choosing $k = m$ and substituting it in the above equation,

$$\begin{aligned}
\mathsf{LCR}_m &\leq \frac{m}{m} + 1 + \frac{g(m) + \frac{m}{m}g(m) - g(m) - m}{\sum_{j=1}^{j=m} v^{(j)} - g(m)}, \\
&\leq 1 + 1 + \frac{g(m) - m}{\sum_{j=1}^{j=m} v^{(j)} - g(m)}, \\
&\overset{(a)}{=} 1 + \frac{\sum_{j=1}^{j=m} v^{(j)} - m}{\sum_{j=1}^{j=m} v^{(j)} - g(m)}, \\
&= 1 + \frac{1}{\frac{\sum_{j=1}^{j=m} v^{(j)} - g(m)}{\sum_{j=1}^{j=m} v^{(j)} - m}}, \\
&= 1 + \frac{1}{1 - \frac{g(m) - m}{\sum_{j=1}^{j=m} v^{(j)} - m}}, \\
&\overset{(b)}{=} 1 + \frac{1}{1 - h(m, g)},
\end{aligned}$$

where (a) follows by adding the second and third terms, and (b) by defining of $h(m,g) = \frac{g(m)-m}{\sum_{j=1}^{j=m} v^{(j)} - m}$.

Next, we prove that $h(m,g) \leq 1/2$, for $\alpha \geq 2$, from which it follows that $\mathsf{LCR}_m \leq 3$. Thus, the competitive ratio of the min-LCR algorithm is at most 3 for all $\alpha > 2$.

From the definition of m, $m = \max_{\{v^{(j)} - [g(j)-g(j-1)]>0\}} j$, all the m highest valued jobs have a value greater than or equal to the effective cost of processing them in slot m, i.e., $v_k \geq c_k = v(k) - v(k-1)$ for $k \leq m$, and hence $\sum_{j=1}^{m} v^{(j)} \geq m[g(m) - g(m-1)]$, thus,

$$h(m,g) \leq \frac{g(m)-m}{m[g(m)-g(m-1)] - m}.$$

Let $\Theta(\alpha) = \dfrac{g(m)-m}{m[g(m)-g(m-1)] - m} = \dfrac{m^\alpha - m}{m[m^\alpha - (m-1)^\alpha] - m}$, since $g(k) = k^\alpha$.

▶ **Lemma 15.** $\forall \alpha \geq 2 \quad \dfrac{\partial \Theta(\alpha)}{\partial \alpha} \leq 0.$

Proof. Taking the derivative and rearranging the terms we get

$$\begin{aligned}
\frac{\partial \Theta(\alpha)}{\partial \alpha} &= -\frac{m^2(m-1)[m^{\alpha-1}((m-1)^{\alpha-1}-1)\log(m) - (m-1)^{\alpha-1}(m^{\alpha-1}-1)\log(m-1)]}{(m[m^\alpha - (m-1)^\alpha] - m)^2}, \\
&\overset{(a)}{=} -\left[\frac{(m^2)(m-1)((m-1)^{\alpha-1}-1)(m^{\alpha-1}-1)}{(\alpha-1)(m[m^\alpha-(m-1)^\alpha]-m)^2}\right] \\
&\quad \cdot \left[\frac{(m^{\alpha-1})\log(m^{\alpha-1})}{(m^{\alpha-1}-1)} - \frac{((m-1)^{\alpha-1})\log((m-1)^{\alpha-1})}{((m-1)^{\alpha-1}-1)}\right], \\
&\overset{(b)}{\leq} 0,
\end{aligned}$$

where (a) follows from writing $\log(m) = \dfrac{\log(m^{\alpha-1})}{\alpha-1}$, and (b) follows from the following three observations i) $\dfrac{x\log(x)}{x-1}$ is an increasing function, ii) $m^{\alpha-1} \geq (m-1)^{\alpha-1}$ and iii) the first term is positive. Note that $\log(m-1)$ is not defined for $m = 1$ but for $m = 1$ observe that $\Theta(\alpha)$ is always 0. Therefore $\forall \alpha \geq 2$ $h(m,g) \leq 0.5$, since for $\alpha = 2$, $\Theta(\alpha) = \dfrac{m^2 - m}{(m^2 - (m-1)^2) - m)} = 0.5$. ◀

◀