Optimality, Fairness, and Robustness in Speed Scaling Designs — Extended Version

Lachlan L.H. Andrew
Centre for Advanced Internet Architectures
Swinburne University of Technology, Australia

Minghong Lin Adam Wierman
Computer Science Department
California Institute of Technology

ABSTRACT

System design must strike a balance between energy and performance by carefully selecting the speed at which the system will run. In this work, we examine fundamental tradeoffs incurred when designing a speed scaler to minimize a weighted sum of expected response time and energy use per job. We prove that a popular dynamic speed scaling algorithm is 2-competitive for this objective and that no "natural" speed scaler can improve on this. Further, we prove that energy-proportional speed scaling works well across two common scheduling policies: Shortest Remaining Processing Time (SRPT) and Processor Sharing (PS). Third, we show that under SRPT and PS, gated-static speed scaling is nearly optimal when the mean workload is known, but that dynamic speed scaling provides robustness against uncertain workloads. Finally, we prove that speed scaling magnifies unfairness, notably SRPT's bias against large jobs and the bias against short jobs in non-preemptive policies. However, PS remains fair under speed scaling. Together, these results show that the speed scalers studied here can achieve any two, but only two, of optimality, fairness, and robustness.

1. INTRODUCTION

Computer systems must make a fundamental tradeoff between performance and energy usage. The days of "faster is better" are gone — energy usage can no longer be ignored in designs, from chips to mobile devices to data centers.

The importance of energy has led designs at all levels of systems to move toward speed scaling, once a technique used primarily at the chip level. Speed scaling designs adapt the "speed" of the system so as to balance energy and performance measures. Speed scaling designs can be highly sophisticated — adapting the speed at all times to the current state (*dynamic speed scaling*) — or very simple — running at a static speed that is chosen *a priori* to balance energy and performance, except when idle (*gated-static speed scaling*).

The growing adoption of speed scaling designs for systems from chips to disks to data centers has spurred analytic research into the topic. The analytic study of the speed scaling problem began with Yao et al. [35] in 1995. Since [35], three main performance objectives balancing energy and delay have been considered: (i) minimize the total energy used in order to meet job deadlines, e.g., [6, 25] (ii) minimize the average response time given an energy/power budget, e.g., [11, 36], and (iii) minimize a linear combination of expected response time and energy usage per job [1, 5]. *In this work we focus on the third objective.* This objective captures how much reduction in response time is necessary to justify using an extra 1 joule of energy, and naturally applies to settings where there is a known monetary cost to extra delay (e.g. many web applications). This is related to (ii) by duality.

Fundamentally, a speed scaling algorithm must make two decisions at each time: (i) a *scheduling policy* must decide which job(s) to service, and (ii) a *speed scaler* must decide how fast to run the server. It has been noted by prior work, e.g., [25], that an optimal speed scaling algorithm will use Shortest Remaining Processing Time (SRPT) scheduling. However, in real systems, it is often impossible to implement SRPT, since it requires exact knowledge of remaining sizes. Instead, typical system designs often use scheduling that is closer to Processor Sharing (PS), e.g., web servers, operating systems, and routers. In this work, we focus on the design of speed scalers for both SRPT and PS.

The study of speed scaling algorithms for these two policies is not new. There has been significant prior work, which we discuss in Sections 3.1 and 3.2, studying speed scaling for SRPT [1, 4, 5, 7, 22] and for PS [10, 14, 16, 30, 32]. Interestingly, the prior work for SRPT is entirely done using a worst-case framework while the prior work for PS is done in a stochastic environment, the M/GI/1 queue.

Despite the considerable literature studying speed scaling, there are many fundamental issues in the design of speed scaling algorithms that are not yet understood. This paper provides new insights into four of these issues:

I *Can a speed scaling algorithm be optimal? What structure do (near-)optimal algorithms have?*
II *How does speed scaling interact with scheduling?*
III *How important is the sophistication of the speed scaler?*
IV *What are the drawbacks of speed scaling?*

To address these questions we study both PS and SRPT scheduling under both dynamic and gated-static speed scaling algorithms. Our work provides (i) new results for dynamic speed scaling with SRPT scheduling in the worst-case model, (ii) the first results for dynamic speed scaling with PS scheduling in the worst-case model, (iii) the first results for dynamic speed scaling with SRPT scheduling in the stochastic model, (iv) the first results for gated-static speed scaling with SRPT in the stochastic model, and (v) the first results identifying unfairness in speed scaling designs. Table 1 summarizes these.

These results lead to important new insights into Issues

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

I-IV above. We describe these insights informally here and provide pointers to the results in the body of the paper.

With respect to **Issue I**, our results show that *"energy-proportional" speed scaling provides near-optimal performance.* Specifically, we consider the algorithm which uses SRPT scheduling and chooses s_n, the speed to run at given n jobs, to satisfy $P(s_n) = n\beta$ (where $P(s)$ is the power needed to run at speed s and $1/\beta$ is the cost of energy). We prove that this algorithm is $(2+\varepsilon)$-competitive under general P (Corollary 3). This provides a tight analysis of an algorithm with a considerable literature, e.g., [1, 4, 5, 7, 22] (see Section 3.1 for a discussion). It also gives analytic justification for a common heuristic applied by system designers, e.g., [8]. Further, we show that no "natural" speed scaling algorithm (Definition 1) can be better than 2-competitive (Theorem 4), which implies that no online energy-proportional speed scaler can match the offline optimal.

With respect to **Issue II**, our results uncover two new insights. First, we prove that, at least with respect to PS and SRPT, *speed scaling can be decoupled from the scheduler.* That is, energy-proportional speed scaling performs well for both SRPT and PS (and another policy LAPS studied in [12]). Specifically, we show that PS scheduling with speeds such that $P(s_n) = n$, which are optimally competitive under SRPT, is again $O(1)$-competitive[1] (Theorem 5). Further, we show that using the speeds optimal for an M/GI/1 PS queue to control instead an M/GI/1 SRPT queue leads to nearly optimal performance (Section 3.2). Second, our results show that scheduling is not as important once energy is considered. Specifically, PS is $O(1)$-competitive for the linear combination of energy and response time; however, when just mean response time is considered PS is $\Omega(\nu^{1/3})$-competitive for instances with ν jobs [24]. Similarly, we see in the stochastic environment that the performance under SRPT and PS is almost indistinguishable (e.g., Figure 1). Together, the insights into Issue II provide a significant simplification of the design of speed scaling systems: they suggest that practitioners can separate two seemingly coupled design decisions and deal with each individually.

With respect to **Issue III**, our results add support to an insight suggested by prior work. Prior work [32] has shown that the optimal gated-static speed scaling algorithm performs nearly as well as the optimal dynamic speed scaling algorithm in the M/GI/1 PS setting. Our results show that the same holds for SRPT (Section 4). Thus, *sophistication does not provide significant performance improvements in speed scaling designs.* However, sophistication provides improved robustness (Section 5). To support this analytically, we provide worst-case guarantees on the (near) optimal stochastic speed scalers for PS and SRPT (Corollary 14). Note that it is rare to be able to provide such guarantees for stochastic control policies. The insights related to Issue III have an interesting practical implication: instead of designing "optimal" speeds it may be better to design "optimally robust" speeds, since the main function of dynamic speed scaling is to provide robustness. This represents a significant shift in approach for stochastic speed scaling design.

With respect to **Issue IV**, our results uncover one unintended drawback of dynamic speed scaling: *speed scaling can magnify unfairness.* Unfairness in speed scaling designs has not been identified previously, but in retrospect the intuition behind it is clear: If a job's size is correlated with the occupancy of the system while it is in service, then dynamic speed scaling will lead to differential service rates across job sizes, and thus unfairness. We prove that speed scaling magnifies unfairness under SRPT (Theorem 16) and all non-preemptive policies, e.g. FCFS (Proposition 17). In contrast, PS is fair even with dynamic speed scaling (Proposition 15). Combining these results with our insights related to Issue II, we see that designers can decouple the scheduler and the speed scaler when considering performance, but should be wary about the interaction when considering fairness.

Our results highlight the balancing act a speed scaling algorithm must perform in order to achieve the three desirable properties: near-optimal performance, robustness, and fairness. It is possible to be near-optimal and robust using SRPT scheduling and dynamic speed scaling, but this creates unfairness. SRPT can be fair and still near-optimal if gated-static speed scaling is used, but this is not robust. On the other hand, dynamic speed scaling with PS can be fair and robust but, in the worst case, pays a significant performance penalty (though in stochastic settings is nearly optimal). Thus, the policies considered in this paper *can achieve any two of near-optimal, fair, and robust — but not all three.*

Finally, it is important to note that the analytic approach of this paper is distinctive. It is unusual to treat both stochastic and worst-case models in one paper; and further, many results depend on a combination of worst-case and stochastic techniques, which leads to insights that could not have been attained by focusing on one model alone.

2. MODEL AND NOTATION

We consider the joint problem of speed scaling and scheduling in a single server queue to minimize a linear combination of expected response time (also called sojourn time or flow time), denoted by T, and energy usage per job, $\mathcal{E}$:

$$z = \mathbb{E}[T] + \mathbb{E}[\mathcal{E}]/\beta. \tag{1}$$

By Little's law, this may be more conveniently expressed as

$$\lambda z = \mathbb{E}[N] + \mathbb{E}[P]/\beta \tag{2}$$

where N is the number of jobs in the system and $P = \lambda\mathcal{E}$ is the power expended.

Before defining the speed scaling algorithms, we need some notation. Let $n(t)$ be the number of jobs in the system at time t and $s(t)$ be the speed that the system is running at time t. Further, define $P(s)$ as the power needed to run at speed s. Then, the energy used by time t is $\mathcal{E}(t) = \int_0^t P(s(\tau))d\tau$.

Measurements have shown that $P(s)$ can take on a variety of forms depending on the system being studied; however, in many applications a low-order polynomial form provides a good approximation, i.e., $P(s) = ks^\alpha$ with $\alpha \in (1,3)$. For example, for dynamic power in CMOS chips $\alpha \approx 1.8$ is a good approximation [32]. However, this polynomial form is not always appropriate; wireless and other communication over an additive white Gaussian noise channel have an exponential power function [13], while interference-limited communications has unbounded power at finite rate [17]. Some of our results assume a polynomial form to make the analysis tractable, and particularly $\alpha = 2$ provides a simple example which we use for many of our numerical experiments. Other results hold for general, even non-convex and discontinuous, power functions. Additionally, we occasionally limit our results to *regular* power functions, which are differentiable on $[0,\infty)$, strictly convex, non-negative, and 0 at speed 0.

Now, we can define a speed scaling algorithm: A speed scaling algorithm $\mathcal{A} = (\pi, \Sigma)$, is a pair of a scheduling dis-

[1] $O(\cdot)$ and $o(\cdot)$ are defined in [9]; $f = \omega(g) \Leftrightarrow g = o(f)$; $f = \Omega(g) \Leftrightarrow g = O(f)$; $f = \Theta(g) \Leftrightarrow [f = O(g)$ and $g = O(f)]$.

Name	Scheduler	Speed scaler: s_n	$P(s)$	Optimal?	Robust?	Fair?
SRPT-INV	SRPT	Dynamic: $P^{-1}(n\beta)$	General	2-competitive (Theorem 1).	yes	no
SRPT-DP	SRPT	Dynamic: Prop. 7	s^α	$O(1)$-competitive for $\alpha \leq 2$ (Corollary 14).	yes	no
SRPT-LIN	SRPT	Dynamic: $n\sqrt{\beta}$	s^2	No guarantee, simulation results in Figure 7.	weakly	no
SRPT-GATED	SRPT	Gated: (22)	Regular	$O(1)$-competitive in M/GI/1 under heavy traffic with $P(s) = s^2$ (Corollary 13). Optimal gated in M/GI/1 under heavy traffic (Theorem 10).	no	yes
PS-INV	PS	Dynamic: $P^{-1}(n\beta)$	s^α	$O(1)$-competitive (Theorem 5).	yes	yes
PS-DP	PS	Dynamic: Prop. 7	s^α	$O(1)$-competitive for $\alpha \leq 2$ (Corollary 14). Optimal in M/GI/1 PS [32].	yes	yes
PS-LIN	PS	Dynamic: $n\sqrt{\beta}$	s^2	$O(1)$-competitive in M/GI/1 with $P(s) = s^2$ [32].	weakly	yes
PS-GATED	PS	Gated: (19)	Regular	$O(1)$-competitive in M/GI/1 with $P(s) = s^2$ (Corollary 13). Optimal gated in M/GI/1 [32].	no	yes

Table 1: Summary of the speed scaling schemes in this paper.

cipline π that defines the order in which jobs are processed, and a speed scaling rule Σ that defines the speed as a function of system state, in terms of the power function, P. In this paper we consider speed scaling rules where the speed is a function of the number of jobs in the system, i.e., s_n is the speed when the occupancy is n.[2]

The scheduling algorithms π we consider are online, and so are not aware of a job j until it arrives at time $r(j)$, at which point π learns the size of the job, x_j. We consider a preempt-resume model, that is, the scheduler may preempt a job and later restart it from the point it was interrupted without any overhead. The policies that we focus on are: Shortest Remaining Processing Time (SRPT), which preemptively serves the job with the least remaining work, and Processor Sharing (PS), which shares the service rate evenly among the jobs in the system at all times.

The speed scaling rules, s_n, we consider can be *gated-static*, which runs at a constant speed while the system is non-idle and sleeps while the system is idle, i.e., $s_n = s_{gs}1_{n\neq 0}$; or more generally *dynamic* $s_n = g(n)$ for some function $g : \mathbb{N} \cup \{0\} \to [0, \infty)$. Note that the speed is simply the rate at which work is completed, i.e., a job of size x served at speed s will complete in time x/s. To avoid confusion, we occasionally write s_n^π as the speed under policy π when the occupancy is n. The queue is single-server in the sense that the full speed s_n can be devoted to a single job.

We analyze the performance of speed scaling algorithms in two different models — one worst-case and one stochastic.

Notation for the worst-case model

In the worst-case model we consider finite, arbitrary (maybe adversarial) deterministic instances of arriving jobs. A problem instance consists of ν jobs, with the jth job having arrival time (release time) $r(j)$ and size (work) x_j. Our objective is again a linear combination of response time and energy usage. Let $\mathcal{E}(I)$ be the total energy used to complete instance I, and T_j be the response time of job j, the completion time minus the release time. The analog of (1) is to replace the ensemble average by the sample average, giving the cost of an instance I under a given algorithm $\mathcal{A}$ as

$$z^{\mathcal{A}}(I) = \frac{1}{\nu}\left(\sum\nolimits_j T_j + \frac{1}{\beta}\mathcal{E}(I)\right). \tag{3}$$

In this model, we compare the cost of speed scaling algorithms to the cost of the optimal offline algorithm, OPT. In particular, we study the competitive ratio, defined as

$$CR = \sup_I z^{\mathcal{A}}(I)/z^O(I),$$

where $z^O(I)$ is the optimal cost achievable on I. A scheme is "c-competitive" if its competitive ratio is at most c.

Notation for the stochastic model

In the stochastic model, we consider an M/GI/1 (or sometimes GI/GI/1) queue with arrival rate λ. Let X denote a random job size with c.d.f. $F(x)$, c.c.d.f. $\bar{F}(x)$, and continuous p.d.f. $f(x)$. Let $\rho = \lambda\mathbb{E}[X] \in [0, \infty)$ denote the load of arriving jobs. Note that ρ is not the utilization of the system and that many dynamic speed scaling algorithms are stable for all ρ. When the power function is $P(s) = s^\alpha$, it is natural to use a scaled load, $\gamma := \rho/\beta^{1/\alpha}$, which jointly characterizes the impact of ρ and β (see [32]).

Denote the response time of a job of size x by $T(x)$. We consider the performance metric (1) where the expectations are averages per job. In this model the goal is to optimize this cost for a specific workload, ρ. Define the competitive ratio in the M/GI/1 model as

$$CR = \sup_{F,\lambda} z^{\mathcal{A}}/z^O$$

where z^O is the average cost of the optimal offline algorithm.

3. DYNAMIC SPEED SCALING

We start by studying the most sophisticated speed scaling algorithms, those that dynamically adjust the speed as a function of the queue length. In this section we investigate the structure of the "optimal" speed scaling algorithm in two ways: (i) we study near-optimal speed scaling rules in the case of both SRPT and PS scheduling; (ii) we study each of these algorithms in both the worst-case model and the stochastic model.

[2]This suits objective (1); e.g., it is optimal for an isolated batch arrival, and the optimal s is constant between arrival/departures. For other objectives, it is better to base the speed on the unfinished work instead [7].

3.1 Worst-case analysis

There has been significant work studying speed scaling in the worst-case model following Yao et al.'s seminal 1995 paper [35], most of it focusing on SRPT. A promising algorithm that has emerged is (SRPT, $P^{-1}(n)$), and there has been a significant stream of papers providing upper bounds on the competitive ratio of this algorithm for objective (1): for unit-size jobs in [1, 7] and for general jobs with $P(s) = s^\alpha$ in [4, 22]. A major breakthrough was made in [5], which shows the 3-competitiveness of $(SRPT, P^{-1}(n+1))$ for general P.

Our contribution to this literature is twofold. First, we tightly characterize the competitive ratio of (SRPT, $P^{-1}(n\beta)$). Specifically, we prove that (SRPT, $P^{-1}(n\beta)$) is exactly 2-competitive under general power functions (see Theorem 1 and Corollary 3). Second, we prove that no "natural" speed scaling algorithm can be better than 2-competitive. Natural speed scaling algorithms include algorithms which have speeds that grow faster, slower, or proportional to $P^{-1}(n\beta)$, or that use a scheduler that works on exactly one job between arrival/departure events (see Definition 1). Thus, the class of natural algorithms includes energy-proportional designs for all schedulers and SRPT scheduling for any s_n. We conjecture that this result can be extended to all speed scaling algorithms, which would imply that the competitive ratio of (SRPT, $P^{-1}(n\beta)$) is minimal.

In contrast to this stream of work studying SRPT, there has been no analysis of speed scaling under PS. We prove that (PS, $P^{-1}(n\beta)$) is $O(1)$-competitive for $P(s) = s^\alpha$ with fixed α, and in particular is $(4\alpha - 2)$-competitive for typical α, i.e., $\alpha \in (1, 3]$. This builds on [12], which studies LAPS, another policy "blind" to job sizes. (LAPS, $P^{-1}(n\beta)$) is also $O(1)$-competitive in this case. For both PS and LAPS the competitive ratio is unbounded for large α, which [12] proves holds for all blind policies. But, note that $\alpha \in (1, 3]$ in most computer systems today (e.g., disks, chips, and servers); thus, asymptotics in α are less important than the performance for small α.

The results in this section highlight important insights about fundamental issues in speed scaling design. First, the competitive ratio results highlight that energy-proportional speed scaling ($P(s_n) = n\beta$) is nearly optimal, which provides analytic justification of a common design heuristic, e.g., [8]. Second, note that energy-proportional speed scaling works well for PS and SRPT (and LAPS). This suggests a designer may decouple the choice of a speed scaler from the choice of a scheduler, choices that initially seem very intertwined. Though we have seen this decoupling only for PS, SRPT, and LAPS, we conjecture that it holds more generally. Third, scheduling seems much less important in the speed scaling model than in the standard constant speed model. For an instance of ν jobs, PS is $\Omega(\nu^{1/3})$-competitive for mean response time in the constant speed model [24], but is $O(1)$-competitive in the speed scaling model. Again, we conjecture that this holds more generally than for just PS.

3.1.1 Amortized competitive analysis

The proofs of the results described above use a technique termed *amortized local competitive analysis* [15, 28]. The technique works as follows.

To show that an algorithm $\mathcal{A}$ is c-competitive with an optimal algorithm OPT for a performance metric $z = \int \zeta(t)dt$ it is sufficient to find a *potential function* $\Phi : \mathbb{R} \to \mathbb{R}$ such that, for any instance of the problem:

1. *Boundary condition:* $\Phi = 0$ before the first job is released, and $\Phi \geq 0$ after the last job is finished;
2. *Jump condition:* At any point where Φ is not differentiable, it does not increase;
3. *Running condition:* When Φ is differentiable,

$$\zeta^{\mathcal{A}}(t) + \frac{d\Phi}{dt} \leq c\zeta^{O}(t), \tag{4}$$

where $\zeta^{\mathcal{A}}(t)$ and $\zeta^{O}(t)$ are the cost $\zeta(t)$ under $\mathcal{A}$ and OPT respectively.

Given these conditions, the competitiveness follows from integrating (4), which gives

$$z^{\mathcal{A}} \leq z^{\mathcal{A}} + \Phi(\infty) - \Phi(-\infty) \leq cz^{O}.$$

3.1.2 SRPT analysis

We now state and prove our results for SRPT.

THEOREM 1. *For any regular power function P, (SRPT, $P^{-1}(n\beta)$) has a competitive ratio of exactly 2.*

The proof of the upper bound is a refinement of the analysis in [5] that accounts more carefully for some boundary cases. It uses the potential function:

$$\Phi(t) = \int_0^\infty \sum_{i=1}^{n[q;t]} \Delta(i)\, dq \tag{5}$$

for some non-decreasing $\Delta(\cdot)$ with $\Delta(i) = 0$ for $i \leq 0$, where $n[q;t] = \max(0, n^A[q;t] - n^O[q;t])$ with $n^A[q;t]$ and $n^O[q;t]$ the number of unfinished jobs at time t with remaining size at least q under the scheme under investigation and the optimal (offline) scheme, respectively.

The following technical lemma is the key step of the proof and is proven in Appendix A.

LEMMA 2. *Let $\eta \geq 1$ and Φ be given by (5) with*

$$\Delta(i) = \frac{1+\eta}{\beta} P'\left(P^{-1}(i\beta)\right). \tag{6}$$

Let $\mathcal{A} = (SRPT, s_n)$ with $s_n \in [P^{-1}(n\beta), P^{-1}(\eta n\beta)]$. Then at points where Φ is differentiable,

$$n^{\mathcal{A}} + P(s^{\mathcal{A}})/\beta + \frac{d\Phi}{dt} \leq (1+\eta)(n^O + P(s^O)/\beta). \tag{7}$$

Using the above Lemma, we can now prove Theorem 1.

PROOF OF THEOREM 1. To show that the competitive ratio of (SRPT, $P^{-1}(n\beta)$) is at most 2, we show that Φ given by (5) and (6) is a valid potential function.

The boundary conditions are satisfied since $\Phi = 0$ when there are no jobs in the system. Also, Φ is differentiable except when a job arrives or departs. When a job arrives, the change in $n^A[q]$ equals that in $n^O[q]$ for all q, and so Φ is unchanged. When a job is completed, $n[q]$ is unchanged for all $q > 0$, and so Φ is again unchanged. The running condition is established by Lemma 2 with $\eta = 1$.

To prove the lower bound on the competitive ratio, consider periodic unit-work arrivals at rate $\lambda = s_n$ for some n. As the number of jobs that arrive grows large, the optimal schedule runs at rate λ, and maintains a queue of at most one packet (the one in service), giving a cost per job of at most $(1 + P(\lambda)/\beta)/\lambda$. In order to run at speed λ, the schedule (SRPT, $P^{-1}(n\beta)$) requires $n = P(\lambda)/\beta$ jobs in the queue, giving a cost per job of $(P(\lambda)+P(\lambda))/(\lambda\beta)$. The competitive ratio is thus at least $\frac{2P(\lambda)}{\beta+P(\lambda)}$. As λ becomes large, this tends to 2 since a regular P is unbounded. □

Theorem 1 can easily be extended to non-negative power functions by applying the same argument as used in [5].

COROLLARY 3. *Let $\varepsilon > 0$. For any non-negative and unbounded $\tilde{P}$, there exists a P such that emulating (SRPT, $P^{-1}(n\beta)$) yields a $(2+\varepsilon)$-competitive algorithm.*

This emulation involves avoiding speeds where P is not convex, instead emulating such speeds by switching between a higher and lower speed on the convex hull of $\tilde{P}$.

Corollary 3 shows that (SRPT, $P^{-1}(n\beta)$) does not match the performance of the offline optimal. This motivates considering other algorithms; however we now show that no "natural" algorithm can do better.

DEFINITION 1. *A speed scaling algorithm $\mathcal{A}$ is **natural** if it runs at speed s_n when it has n unfinished jobs, and for convex P, one of the following holds:*

(a) the scheduler is work-conserving and works on a single job between arrival/departure events; or
(b) $g(s) + P(s)/\beta$ is convex, for some g with $g(s_n) = n$; or
(c) the speeds s_n satisfy $P(s_n) = \omega(n)$; or
(d) the speeds s_n satisfy $P(s_n) = o(n)$.

This fragmented definition seems "unnatural", but the class contains most natural contenders for optimality: all algorithms that use the optimal scheduler SRPT, and all whose speeds grow faster than, slower than, or proportional to $P^{-1}(n)$. To be "unnatural", an algorithm must have speeds which increase erratically (or decrease) as n increases.

THEOREM 4. *For any $\varepsilon > 0$ there is a regular power function P_ε such that any natural algorithm $\mathcal{A}$ on P_ε has competitive ratio larger than $2 - \varepsilon$.*

This theorem highlights that if an algorithm does have a smaller competitive ratio than (SRPT, $P^{-1}(n\beta)$), it will not use "natural" scheduling or speed scaling. Though the result only applies to natural algorithms, we conjecture that, in fact, it holds for all speed scaling algorithms, and thus the competitive ratio of (SRPT, $P^{-1}(n\beta)$) is minimal.

PROOF. Consider the case when $P(s) = s^\alpha$, with α yet to be determined. We show that, for large α, the competitive ratio is at least $2-\varepsilon$, by considering two cases: instance $I_{B(\nu)}$ is a *batch arrival* of ν jobs of size 1 at time 0 with no future arrivals, and instance $I_{R(b,\lambda)}$ is a batch of b jobs at time 0 followed by a long train of *periodic arrivals* of jobs of size 1 at times k/λ for $k \in \mathbb{N}$.

Fix an $\varepsilon > 0$ and consider a speed scaling which can attain a competitive ratio of $2 - \varepsilon$ for all instances $I_{R(\cdot,\cdot)}$. For $I_{R(\cdot,\lambda)}$, with large λ, the optimal algorithm will run at speed exceeding λ for a finite time until the occupancy is one. After that, it will run at speed λ so that no queue forms. For long trains, this leads to a cost per job of $(1 + P(\lambda)/\beta)/\lambda$.

First, consider a "type (d)" natural $\mathcal{A}$. For sufficiently large λ, $n > ks_n^\alpha$ for all $s_n \geq \lambda/2$, where $k = 2^{\alpha+2}/\beta$. Between arrivals, at least $1/2$ unit of work must be done at speed at least $\lambda/2$, in order for $\mathcal{A}$ not to fall behind. The cost per unit work is at least $(1/s)(ks^\alpha + s^\alpha/\beta)$, and so the total cost of performing this $1/2$ unit is at least $(k + 1/\beta)\lambda^{\alpha-1}/2^\alpha > 4\lambda^{\alpha-1}/\beta$. For large λ, this is at least twice the cost per job under the optimal scheme: $(1 + P(\lambda)/\beta)/\lambda < 2\lambda^{\alpha-1}/\beta$.

It remains to consider natural algorithms of types (a)–(c).

Consider a "type (a)" natural $\mathcal{A}$ on the instance $I_{R(n,s_n)}$ for some n. It will initially process exactly one job at speed s_n, which it will finish at time $1/s_n$. From this time, a new arrival will occur whenever a job completes, and so the algorithm runs at speed s_n with occupancy n until the last arrival. So, the average cost per job tends to $(n+P(s_n)/\beta)/s_n$ on large instances, leading to a competitive ratio of:

$$1 + \frac{n-1}{P(s_n)/\beta + 1} \leq CR_{periodic} \leq 2 - \varepsilon. \tag{8}$$

Consider a "type (b)" natural $\mathcal{A}$. On $I_{R(n,s_n)}$, $\mathcal{A}$ also satisfies (8): Let $\bar{s}$ denote the time-average speed. For all $\phi < 1$, for sufficiently long instances we need $\bar{s} \geq \phi s_n$ to prevent an unbounded queue forming. By Jensen's inequality, the average cost per job satisfies $\bar{z} \geq (g(\bar{s}) + P(\bar{s})/\beta) \geq (g(\phi s_n) + P(\phi s_n)/\beta)$. Since ϕ can be arbitrarily close to 1, the cost can be arbitrarily close to $n+P(s_n)/\beta$, implying (8).

For a "type (c)" natural $\mathcal{A}$, $P(s_n)/n \to \infty$ for large n.

Thus, for types (a)–(c), $\exists n_0$ such that for all $n > n_0$:

$$s_n \geq \hat{s}_n := P^{-1}\left(\frac{n\beta}{1-\varepsilon/2}\right). \tag{9}$$

We now show that this condition precludes having a competitive ratio of $2 - \varepsilon$ in the case of batch arrivals, $I_{B(\nu)}$.

For $I_{B(\nu)}$, the optimal strategy is to server one job at a time at some speeds s_n^*, giving cost

$$\begin{aligned} z^O(I_{B(\nu)}) &= \sum_{n=1}^{\nu} \frac{n}{s_n^*} + \frac{P(s_n^*)}{\beta s_n^*} \\ &= \sum_{n=1}^{\nu} \frac{n^{(\alpha-1)/\alpha}}{\beta^{1/\alpha}} \left[\left(\frac{n\beta}{(s_n^*)^\alpha}\right)^{1/\alpha} + \left(\frac{(s_n^*)^\alpha}{n\beta}\right)^{(\alpha-1)/\alpha}\right]. \end{aligned}$$

The unique local minimum of $\phi(\cdot) = (\cdot)^{(\alpha-1)/\alpha} + (\cdot)^{-1/\alpha}$ occurs at $1/(\alpha-1)$. This gives a minimum cost of

$$z^O(I_{B(\nu)}) = \frac{\alpha \sum_{n=1}^{\nu} n^{(\alpha-1)/\alpha}}{\beta^{1/\alpha}(\alpha-1)^{(\alpha-1)/\alpha}}$$

for $s_n^* = (n\beta/(\alpha-1))^{1/\alpha}$. More generally, the optimum is

$$\beta n = s_n^* P'(s_n^*) - P(s_n^*). \tag{10}$$

Under $\mathcal{A}$, when more than $n-1$ work remains, there must be at least n unfinished jobs. Thus, for $\alpha - 1 > 1 - \varepsilon/2$,

$$z(I_{B(\nu)}) \geq \sum_{n=n_0}^{\nu} \frac{n^{(\alpha-1)/\alpha}}{\beta^{1/\alpha}} \left[\left(\frac{n\beta}{(\hat{s}_n)^\alpha}\right)^{1/\alpha} + \left(\frac{(\hat{s}_n)^\alpha}{n\beta}\right)^{(\alpha-1)/\alpha}\right].$$

since the minimum of $\phi(\cdot)$ subject to (9) then occurs at $(\hat{s}_n)^\alpha/(n\beta)$. Since $(\hat{s}_n)^\alpha/(n\beta) = 1/(1-\varepsilon/2)$, this gives

$$\begin{aligned} CR_{batch} \geq &\left(\frac{\sum_{n=n_0}^{\nu} n^{(\alpha-1)/\alpha}}{\sum_{n=1}^{\nu} n^{(\alpha-1)/\alpha}}\right)\left(\frac{(\alpha-1)^{(\alpha-1)/\alpha}}{\alpha}\right) \\ &\left[\left(\frac{1}{1-\varepsilon/2}\right)^{(\alpha-1)/\alpha} + \left(\frac{1}{1-\varepsilon/2}\right)^{-1/\alpha}\right]. \end{aligned}$$

For any $\varepsilon \in (0,1)$, the product of the last two factors tends to $1+1/(1-\varepsilon/2)$ as $\alpha \to \infty$, and hence there is an $\alpha = \alpha(\varepsilon)$ for which their product exceeds $1/(1-\varepsilon/3)+1$. Similarly, for all $\alpha > 1$, there is a sufficiently large ν that the first factor exceeds $1/(1+\varepsilon/9)$. For this α and ν, $CR_{batch} > 2$.

So, for $P(s) = s^{\alpha(\varepsilon)}$, if the competitive ratio is smaller than $2 - \varepsilon$ in the periodic case, it must be larger than 2 in the batch case. □

Theorem 4 relies on P being highly convex, as in interference-limited systems [17]. For CMOS systems in which typically $\alpha \in (1, 3]$, it is possible to design natural algorithms that can outperform (SRPT, $P^{-1}(n\beta)$).

3.1.3 PS analysis

We now state and prove our bound on the competitive ratio of PS.

THEOREM 5. *If $P(s) = s^\alpha$ then (PS, $P^{-1}(n\beta)$) is $\max(4\alpha - 2, 2(2 - 1/\alpha)^\alpha)$-competitive.*

In particular, PS is $(4\alpha-2)$-competitive for α in the typical range of $(1, 3]$.

Theorem 5 is proven using amortized local competitiveness. Let $\eta \geq 1$, and $\Gamma = (1+\eta)(2\alpha-1)/\beta^{1/\alpha}$. The potential function is then defined as

$$\Phi = \Gamma \sum_{i=1}^{n^A(t)} i^{1-1/\alpha} \max(0, q^A(j_i;t) - q^O(j_i;t)) \qquad (11)$$

where $q^\pi(j;t)$ is the remaining work on job j at time t under scheme π, and $\{j_i\}_{i=1}^{n^A(t)}$ is an ordering of the jobs in increasing order of release time: $r(j_1) \leq r(j_2) \leq \cdots \leq r(j_{n^A(t)})$. Note that this is a scaling of the potential function that was used in [12] to analyze LAPS. As a result, to prove Theorem 5, we can use the corresponding results in [12] to verify the boundary and jump conditions. All that remains is the running condition, which follows from the technical lemma below. The proof is provided in Appendix B.

LEMMA 6. *Let Φ be given by (11) and A be the discipline (PS, s_n) with $s_n \in [(n\beta)^{1/\alpha}, (\eta n\beta)^{1/\alpha}]$. Then under A, at points where Φ is differentiable,*

$$n^A + (s^A)^\alpha/\beta + \frac{d\Phi}{dt} \leq c(n^O + (s^O)^\alpha/\beta) \qquad (12)$$

where $c = (1+\eta)\max((2\alpha-1), (2-1/\alpha)^\alpha)$.

3.2 Stochastic analysis

We now study optimal dynamic speed scaling in the stochastic setting. In contrast to the worst-case results, in the stochastic setting, it is possible to optimize the algorithm for the expected workload. In a real application, it is clear that incorporating knowledge about the workload into the design can lead to improved performance. Of course, the drawback is that there is always uncertainty about workload information, either due to time-varying workloads, measurement noise, or simply model inaccuracies. We discuss robustness to these factors in Section 5, and in the current section assume that exact workload information is known to the speed scaler and that the model is accurate.

In this setting, there has been a substantial amount of work studying the M/GI/1 PS model [10, 14, 16, 30][3]. This work is in the context of operations management and so focuses on "operating costs" rather than "energy", but the model structure is equivalent. This series of work formulates the determination of the optimal speeds as a stochastic dynamic programming (DP) problem and provides numeric techniques for determining the optimal speeds, as well as proving that the optimal speeds are monotonic in the queue length. The optimal speeds have been characterized as follows [32]. Recall that $\gamma = \rho/\beta^{1/\alpha}$.

PROPOSITION 7. *Consider an M/GI/1 PS queue with controllable service rates s_n. Let $P(s) = s^\alpha$. The optimal dynamic speeds are concave and satisfy the dynamic program given in [32]. For $\alpha = 2$ and any $n \geq 2\gamma$, they satisfy*

$$\gamma + \sqrt{n - 2\gamma} \leq \frac{s_n}{\sqrt{\beta}} \leq \gamma + \sqrt{n} + \min\left(\frac{\gamma}{2n}, \gamma^{1/3}\right). \qquad (13)$$

For general $\alpha > 1$, they satisfy[4]

$$\frac{s_n}{\beta^{1/\alpha}} \leq \left(\frac{1}{\alpha}\min_{\sigma>\gamma}\left(\frac{n+\sigma^\alpha-\gamma^\alpha}{(\sigma-\gamma)} + \frac{\gamma}{(\sigma-\gamma)^2}\right)\right)^{1/(\alpha-1)} \qquad (14)$$

$$\frac{s_n}{\beta^{1/\alpha}} \geq \left(\frac{n}{\alpha-1}\right)^{1/\alpha}. \qquad (15)$$

PROOF. Bounds (13) and (14) are shown in [32]. Additionally, the concavity of s_n follows from results in [32]. To prove (15), note that when $\rho = 0$ the optimal speeds are those optimal for batch arrivals, which satisfy (15) by (10). Then, it is straightforward from the DP that s_n increases monotonically with load ρ, which gives (15). □

Interestingly, the bounds in Proposition 7 are tight for large n and have a form similar to the form of the worst-case speeds for SRPT and PS in Theorems 1 and 5.

In contrast to the large body of work studying the optimal speeds under PS scheduling, there is no work characterizing the optimal speeds under SRPT scheduling. This is not unexpected since the analysis of SRPT in the static speed setting is significantly more involved than that of PS. Thus, instead of analytically determining the optimal speeds for SRPT, we are left to use a heuristic approach.

Note that the speeds suggested by the worst-case results for SRPT and PS (Theorems 1 and 5) are the same, and the optimal speeds for a batch arrival are given by (10) for both policies. Motivated by this and the fact that (10) matches the asymptotic form of the stochastic results for PS in Proposition 7, *we propose to use the optimal PS speeds in the case of SRPT.*

To evaluate the performance of this heuristic, we use simulation experiments (Figure 1) that compare the performance of this speed scaling algorithm to the following lower bound.

PROPOSITION 8. *In a GI/GI/1 queue with $P(s) = s^\alpha$,*

$$z^O \geq \frac{1}{\lambda}\max(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}).$$

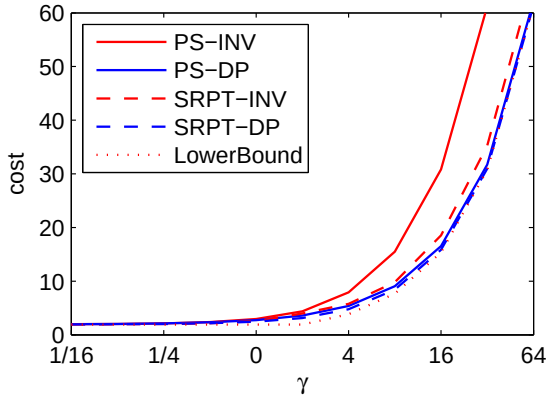
This was proven in [32] in the context of the M/GI/1 PS but the proof can easily be seen to hold more generally.

Simulation experiments also allow us to study other interesting topics, such as (i) a comparison of the performance of the worst-case schemes for SRPT and PS with the stochastic schemes and (ii) a comparison of the performance of SRPT and PS in the speed scaling model. In these experiments, the optimal speeds for PS in the stochastic model are found using the numeric algorithm for solving the DP described in [16, 32], and then these speeds are also used for SRPT. For brevity, we describe the results from only one of many settings we investigated.

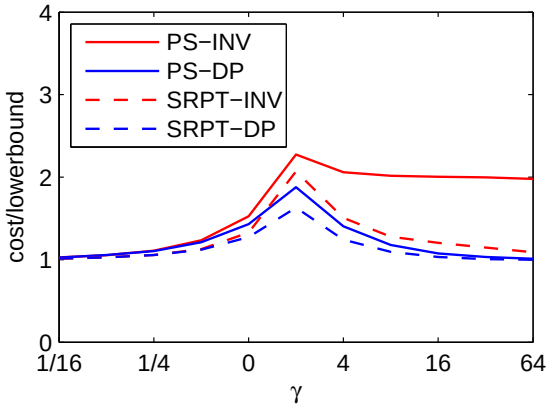
Figure 2 shows that the optimal speeds from the DP ("DP") have a similar form to the speeds motivated by the worst-case results, $P^{-1}(n\beta)$ ("INV"), differing by γ for high queue occupancies. Figure 1 shows how the total cost (1) depends on the choice of speeds and scheduler. At low loads, all schemes are indistinguishable. At higher loads, the performance of the PS-INV scheme degrades significantly, but the SRPT-INV scheme maintains fairly good performance. Note though that if $P(s) = s^\alpha$ for $\alpha > 3$ the performance of SRPT-INV degrades significantly too. In contrast, the DP-based schemes benefit significantly from having the slightly higher speeds chosen to optimize (1) rather than minimize the competitive ratio. Finally, the SRPT-DP scheme performs nearly optimally, which justifies the heuristic of using

[3]These actually study the M/M/1 FCFS queue, but since the M/GI/1 PS queue with controllable service rates is a symmetric discipline [19] it has the same occupancy distribution and mean delay as an M/M/1 FCFS queue.

[4]In [32] the range of minimization was misstated as $\sigma > 0$.



(a)



(b)

Figure 1: Comparison of SRPT and PS scheduling under both $s_n = P^{-1}(n\beta)$ and speeds optimized for an M/GI/1 PS system, using Pareto(2.2) job sizes and $P(s) = s^2$.

the optimal speeds for PS in the case of SRPT[5]. However, the PS-DP scheme performs nearly as well as SRPT-DP. Together, these observations suggest that it is important to optimize the speed scaler, but not necessarily the scheduler.

4. GATED-STATIC SPEED SCALING

Section 3 studied a sophisticated form of speed scaling where the speed can depend on the current occupancy. This scheme can perform (nearly) optimally; however its complexity and overheads may be prohibitive. This is in contrast to the simplest non-trivial form: *gated-static* speed scaling, where $s_n = s_{gs} 1_{n\neq 0}$ for some constant speed s_{gs}. This requires minimal hardware to support; e.g., a CMOS chip may have a constant clock speed but AND it with the gating signal to set the speed to 0.

Gated-static speed scaling can be arbitrarily bad in the worst case since jobs can arrive faster than s_{gs}. Thus, we study gated-static speed scaling only in the stochastic model, where the constant speed s_{gs} can depend on the load.

We study the gated-static speed scaling under SRPT and PS scheduling. The optimal gated-static speed under PS has been derived in [32], but the optimal speed under SRPT has not been studied previously.

Our results highlight two practical insights. First, we show that gated-static speed scaling can provide nearly the same cost as the optimal dynamic policy in the stochastic model. Thus, the simplest policy can nearly match the performance of the most sophisticated policy. Second, we show that the performance of gated-static under PS and SRPT is not too different, thus scheduling is much less important to optimize than in systems in which the speed is fixed in advance. This reinforces what we observed for dynamic speed scaling.

4.1 Optimal gated-static speeds

We now derive the optimal speed s_{gs}, which minimizes the expected cost of gated-static in the stochastic model under both SRPT and PS. First note that, since the power cost is constant at $P(s_{gs})$ whenever the server is running, the optimal speed is

$$s_{gs} = \arg\min_s \beta\mathbb{E}[T] + \frac{1}{\lambda}P(s)\Pr(N \neq 0). \tag{16}$$

In the second term $\Pr(N \neq 0) = \rho/s$, and so multiplying by λ and setting the derivative to 0 gives that the optimal gated-static speed satisfies

$$\beta\frac{d\mathbb{E}[N]}{ds} + r\frac{P^*(s)}{s} = 0, \tag{17}$$

where $r = \rho/s$ is the utilization and

$$P^*(s) \equiv sP'(s) - P(s). \tag{18}$$

Note that if P is convex then P^* is increasing and if P'' is bounded away from 0 then P^* is unbounded.

Under PS, $\mathbb{E}[N] = \rho/(s-\rho)$, and so $d\mathbb{E}[N]/ds = \mathbb{E}[N]/(\rho - s)$. By (17), the optimal speeds satisfy [32]

$$\beta\mathbb{E}[N] = (1-r)rP^*(s). \tag{19}$$

Unfortunately, in the case of SRPT, things are not as easy. For $s = 1$, it is well known, e.g., [21], that

$$\mathbb{E}[T] = \int_{x=0}^{\infty}\int_{t=0}^{x} \frac{dt}{1-\lambda\int_0^t \tau\, dF(\tau)} + \frac{\lambda\int_0^x \tau^2\, dF(\tau) + x^2\bar{F}(x)}{2(1-\lambda\int_0^x \tau dF(\tau))^2}\, dF(x)$$

The complexity of this equation rules out calculating the speeds analytically. So, instead we use simpler forms for $\mathbb{E}[N]$ that are exact in asymptotically heavy or light traffic.

4.1.1 A heavy-traffic approximation

We state the heavy-traffic results for distributions whose c.c.d.f. $\bar{F}$ has lower and upper Matuszewska indices [9] of m and M. Intuitively, $C_1x^m \lesssim \bar{F}(x) \lesssim C_2x^M$ as $x \to \infty$ for some C_1, C_2. So, the Matuszewska index can be thought of as a "moment index." Further, let $G(x) = \int_0^x tf(t)\,dt/\mathbb{E}[X]$ be the fraction of work coming from jobs of size at most x. The following was proven in [23].

PROPOSITION 9 ([23]). *For an $M/GI/1$ under SRPT with speed 1, $\mathbb{E}[N] = \Theta(H(\rho))$ as $\rho \to 1$, where*

$$H(\rho) = \begin{cases} E[X^2]/((1-\rho)G^{-1}(\rho)) & \text{if } M < -2 \\ E[X]\log(1/(1-\rho)) & \text{if } m > -2. \end{cases} \tag{20}$$

Proposition 9 motivates the following heavy-traffic approximation for the case when the speed is 1:

$$\mathbb{E}[N] \approx CH(\rho) \tag{21}$$

where C is a constant dependent on the job size distribution. For job sizes which are Pareto(a) (or more generally, regularly varying [9]) with $a > 2$, it is known that

[5]Note that the peak around $\gamma = 1$ in Fig. 1(b) is most likely due to the looseness of the lower bound.

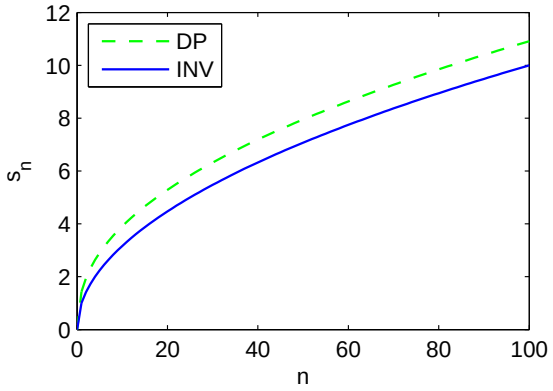


Figure 2: Comparison of $s_n = P^{-1}(n\beta)$ with speeds "DP" optimized for an M/GI/1 system with $\gamma = 1$ and $P(s) = s^2$.

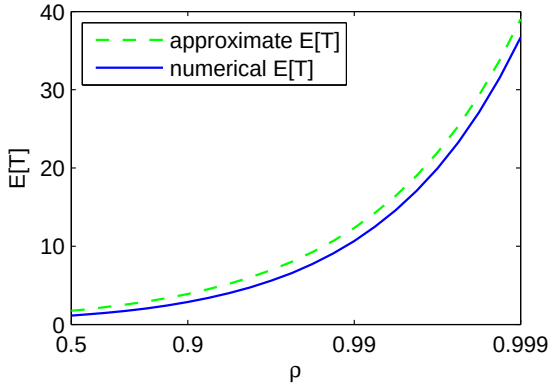


Figure 3: Validation of the heavy-traffic approximation (21) by simulation using Pareto(3) job sizes with $E[X] = 1$.

$C = (\pi/(1-a))/(2\sin(\pi/(1-a)))$ [23]. Figure 3 shows that in this case, the heavy-traffic results are accurate even for quite low loads.

Given approximation (21), we can now return to equation (17) and calculate the optimal speed for gated-static SRPT. Define $h(r) = (G^{-1})'(r)/G^{-1}(r)$.

THEOREM 10. *Suppose approximation (21) holds with equality.*

(i) If $M < -2$, then for the optimal gated-static speed,

$$\beta\mathbb{E}[N]\left(\frac{2-r}{1-r} - rh(r)\right) = rP^*(s). \tag{22a}$$

(ii) If $m > -2$, then for the optimal gated-static speed,

$$\beta\mathbb{E}[N]\left(\frac{1}{(1-r)\log(1/(1-r))}\right) = P^*(s). \tag{22b}$$

PROOF. If $M < -2$, let C be such that

$$\mathbb{E}[N] = \frac{C\mathbb{E}[X^2]/s^2}{(1-\rho/s)G^{-1}(\rho/s)} \tag{23}$$

Then

$$\begin{aligned}\frac{d\mathbb{E}[N]}{ds} &= \frac{\lambda C\mathbb{E}[X^2]\left((\rho-2s)sG^{-1}(\frac{\rho}{s}) + \rho(s-\rho)(G^{-1})'(\frac{\rho}{s})\right)}{(s-\rho)^2 s^3 (G^{-1}(\rho/s))^2} \\ &= \frac{\mathbb{E}[N]}{s}\left(\frac{\rho/s-2}{1-\rho/s} + \frac{\rho(G^{-1})'(\rho/s)}{sG^{-1}(\rho/s)}\right)\end{aligned} \tag{24}$$

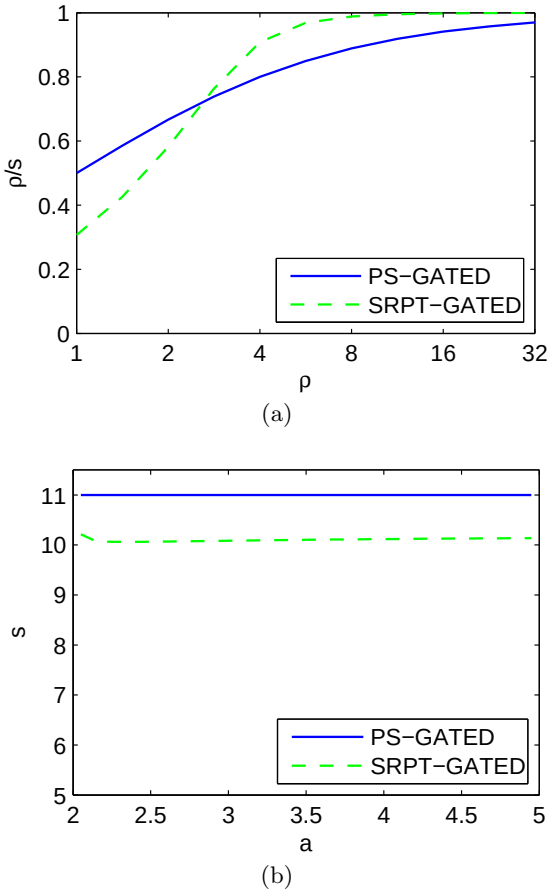


Figure 4: Comparison for gated-static: PS using (19) and SRPT using (27), with $P(s) = s^2$. (a) Utilization given Pareto(2.2) job sizes. (b) Dependence of speed on the job size distribution, for Pareto(a).

Substituting this into (17) gives (22a).

If $m > -2$, then there is a $C' = CE[X]$ such that

$$\mathbb{E}[N] = \frac{C'}{s}\log\left(\frac{1}{1-\rho/s}\right). \tag{25}$$

for speed s. Now

$$\begin{aligned}\frac{d\mathbb{E}[N]}{ds} &= -\frac{C'}{s^2}\log\left(\frac{1}{1-\rho/s}\right) - \frac{C'\rho}{s^2(s-\rho)} \\ &= -\frac{\mathbb{E}[N]}{s}\left(1 + \frac{\rho}{s}\frac{1}{(1-\rho/s)\log(1/(1-\rho/s))}\right),\end{aligned}$$

and the factor in brackets is dominated by its second term in heavy traffic. Substituting this into (17) gives the result. □

To evaluate the speeds derived for heavy-traffic, Figure 4(b) illustrates the gated-static speeds derived for SRPT and PS, for $P(s) = s^2$ and $\rho = 10$ and varying job size distribution. This suggests that the SRPT speeds are nearly independent of the job size distribution. (Note that the vertical axis does not start from 0.) Moreover, the speeds of SRPT and PS differ significantly in this setting since the speeds under SRPT are approximately minimal (the speeds must be larger than γ), while the PS speeds are $\gamma + 1$.

Theorem 10 assumes that the system is in heavy-traffic. To understand when this holds, first note that if there is a

maximum allowable speed $s_{\max}$ then the heavy-traffic regime is valid as $\rho \uparrow s_{\max}$. In the case when there is no maximum allowable speed, the following applies.

PROPOSITION 11. *If $P^*(s)$ is unbounded as $s \to \infty$ and $-2 \notin [m, M]$ then as $\rho \to \infty$, (22) induces the heavy-traffic regime, $\rho/s \to 1$.*

PROOF. As $s \geq \rho$, we have $s \to \infty$. Thus if $m \leq -2$ then by (22a)

$$\frac{\beta \mathbb{E}[N]}{r}\left(\frac{1}{1-r} + 1 - rh(r)\right) = P^*(s) \to \infty \qquad (26)$$

which implies $r \to 1$, since $\mathbb{E}[N]$ is bounded if r is bounded away from 1. Hence heavy traffic occurs for $\rho \to \infty$.

Conversely, if $M \geq -2$ then the coefficient of $\mathbb{E}[N]$ in (22b) must be unbounded, whence $r \to 1$, and again the heavy-traffic regime occurs for large ρ. □

Figure 4(a) illustrates the effect of raising the load on the utilization in the case of $P(s) = s^2$ and Pareto(2.2) job sizes.

4.1.2 *Beyond heavy-traffic*

Let us next briefly consider the light-traffic regime. As $\rho \to 0$, there is seldom more than one job in the system, and SRPT and PS have nearly indistinguishable $\mathbb{E}[N]$. So, in this case, it is appropriate to use speeds given by (19).

Given the light- and heavy-traffic approximations we have just described, it remains to decide the speed in the intermediate regime. We propose setting

$$s_{gs}^{SRPT} = \min(s_{gs}^{PS}, s_{gs}^{SRPT(HT)}), \qquad (27)$$

where s_{gs}^{PS} satisfies (19), and $s_{gs}^{SRPT(HT)}$ is given by (22) with $\mathbb{E}[N]$ estimated by (21).

To see why (27) is reasonable, we first show that (22) often tends to the optimal speed as $\rho \to 0$.

PROPOSITION 12. *If $m > -2$ or both $M < -2$ and arbitrarily small jobs are possible (i.e., for all $x > 0$ there is a $y \in [0, x]$ with $F(y) > 0$), then (22) produces the optimal scaling as $\rho \to 0$.*

PROOF. For $\rho \to 0$, also $r \to 0$, and $\mathbb{E}[N]/r \to 1$. By L'Hospital's rule $(1-r)\log(1/(1-r))/r \sim 1$, and (22b) gives $\beta = P^*(s)$. If arbitrarily small jobs are possible, then $G^{-1}(0) = 0$, and $rh(r) \to 1$ by L'Hospital's rule, whence (22a) also becomes $\beta = P^*(s)$.

From (10), this is the optimal speed at which to server a batch of a single job. Since, as $\rho \to 0$, the system almost certainly has a single job when it is non-empty, this is an appropriate speed. □

Although (22) tends to the optimal speeds, (21) over estimates $\mathbb{E}[N]$ for small ρ and so $s_{gs}^{SRPT(HT)}$ is higher than optimal for small loads. Conversely, for a given speed, the delay is less under SRPT than PS, and so the optimal speed under SRPT will be lower than that under PS. Hence $s_{gs}^{SRPT(HT)} < s_{gs}^{PS}$ in the large ρ regime where the former becomes accurate. Thus, the min operation in (27) selects the appropriate form in each regime.

4.2 Gated-static vs. dynamic speed scaling

Now that we have derived the optimal gated-static speeds, we can contrast the performance of gated-static with that of dynamic speed scaling. This is a comparison of the most and least sophisticated forms of speed scaling.

As Figure 5 shows, the performance (in terms of mean delay plus mean energy) of a well-tuned gated-static system

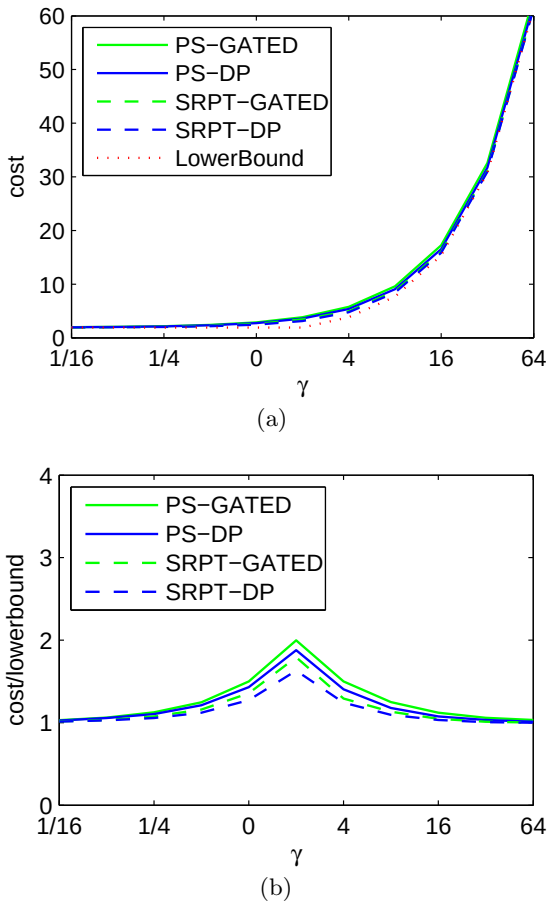


Figure 5: Comparison of PS and SRPT with gated-static speeds (19) and (27), versus the dynamic speeds optimal for an M/GI/1 PS. Job sizes are distributed as Pareto(2.2) and $P(s) = s^2$.

is almost indistinguishable from that of the optimal dynamic speeds. Moreover, there is little difference between the cost under PS-GATED and SRPT-GATED, again highlighting that the importance of scheduling in the speed scaling model is considerably less than in standard queueing models.

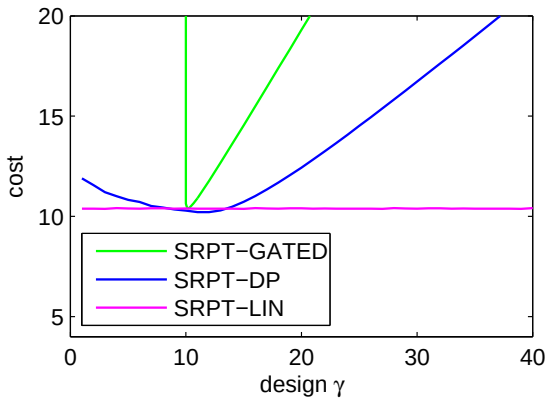
In addition to observing numerically that the gated-static schemes are near optimal, it is possible to provide some analytic support for this fact as well. In [32] it was proven that PS-GATED is within a factor of 2 of PS-DP when $P(s) = s^2$. Combining this result with the competitive ratio results in this paper, we have

COROLLARY 13. *Consider $P(s) = s^2$. The optimal PS and SRPT gated-static designs are $O(1)$-competitive in an M/GI/1 queue with load ρ.*

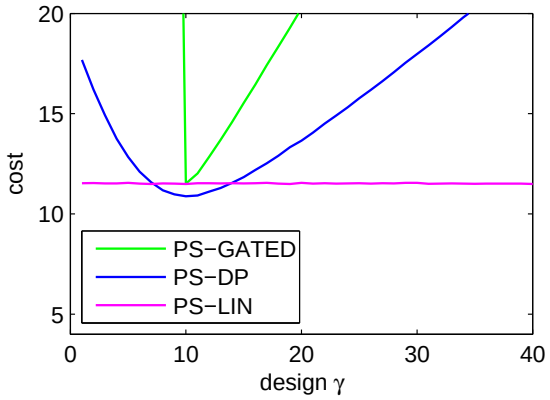
PROOF. Let $\pi \in \{PS, SRPT\}$ and s_{gs}^{π} be the optimal gated-static speed for π and s_n^{DP} be the optimal speeds, which solve the DP for the M/GI/1 PS queue. Then

$$z^{(\pi, s_{gs}^{\pi})} \leq z^{(PS, s_{gs}^{PS})} \leq 2z^{(PS, s_n^{DP})} \leq 2z^{(PS, P^{-1}(n\beta))} \leq 12z^{O}.$$

The last three steps follow from [32], the optimality of DP for PS in M/GI/1, and Theorem 5. □



(a) SRPT



(b) PS

Figure 6: Effect of misestimating γ under PS and SRPT: cost when $\gamma = 10$, but s_n are optimal for a different "design γ". Pareto(2.2) job sizes; $P(s) = s^2$.

5. ROBUSTNESS AND SPEED SCALING

Section 4 shows that near-optimal performance can be obtained using the simplest form of speed scaling — running at a static speed when not idle. Why then do CPU manufacturers design chips with multiple speeds? The reason is that the optimal gated-static design depends intimately on the load ρ. This cannot be known exactly in advance, especially since workloads typically vary over time. So, an important property of a speed scaling design is *robustness* to uncertainty in the workload, ρ and F, and to model inaccuracies.

Figure 6 illustrates that if a gated-static design is used, performance degrades dramatically when ρ is mispredicted. If the static speed is chosen and the load is lower than expected, excess energy will be used. Underestimating the load is even worse; if the system has static speed s and $\rho \geq s$ then the cost is unbounded.

In contrast, Figure 6 illustrates simulation experiments which show that dynamic speed scaling (SRPT-DP) is significantly more robust to misprediction of the workload. In fact, we can prove this analytically by providing worst-case guarantees for the SRPT-DP and PS-DP. Let s_n^{DP} denote the speeds used for SRPT-DP and PS-DP. Note that the corollary below is distinctive in that it provides worst-case guarantees for a stochastic control policy.

COROLLARY 14. *Consider $P(s) = s^\alpha$ with[6] $\alpha \in (1, 2]$ and algorithm $\mathcal{A}$ which chooses speeds s_n^{DP} optimal for PS scheduling in an M/GI/1 queue with load ρ. If $\mathcal{A}$ uses either PS or SRPT scheduling, then $\mathcal{A}$ is $O(1)$-competitive in the worst-case model.*

PROOF. The proof applies Lemmas 2 and 6 from the worst-case model to the speeds from the stochastic model.

By (15) of Proposition 7, $s_n \geq (n\beta/(\alpha-1))^{1/\alpha}$. Since $\alpha < 2$, this implies $s_n \geq P^{-1}(n\beta)$. Further, (14) implies that $s_n^{DP} = O(n^{1/\alpha})$ for any fixed ρ and β and is bounded for finite n.

Hence the speeds s_n^{DP} are of the form given in Lemmas 2 and 6 for some finite η (dependent on π and the constant ρ), from which it follows that $\mathcal{A}$ is constant competitive. □

For $\alpha = 2$, Proposition 7 implies $s_n^{DP} \leq (2\gamma+1)P^{-1}(n\beta)$, whence (SRPT, s_n^{DP}) is $(2\gamma+2)$-competitive.

Corollary 14 highlights that s_n^{DP} designed for a given ρ leads to a speed scaler that is "robust". However, the cost still degrades significantly when ρ is mispredicted badly (as shown in Figure 6).

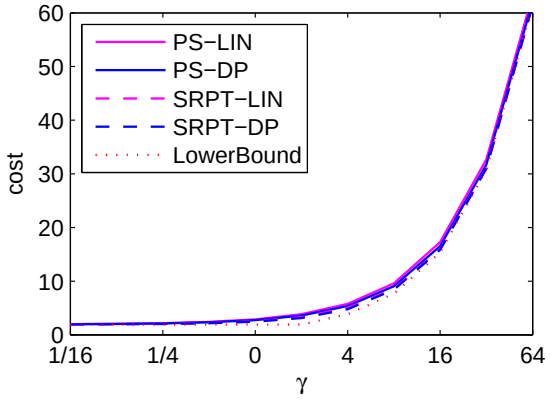
We now consider a different form of robustness: If the arrivals are known to be well approximated by a Poisson process, but ρ is unknown, is it possible to choose speeds that are close to optimal for all ρ? It was shown in [32] that using "linear" speeds, $s_n = n\sqrt{\beta}$, gives near-optimal performance when $P(s) = s^2$ and PS scheduling is used. This scheme ("LIN") performs much better than using $s_n = P^{-1}(n\beta)$, despite the fact that it also uses no knowledge of the workload. Given the decoupling of scheduling and speed scaling suggested by the results in Section 3, this motivates using the same linear speed scaling for SRPT. Figure 7 illustrates that this linear speed scaling provides near-optimal performance under SRPT too. The robustness of this speed scaling is illustrated in Figure 6. However, despite being more robust in the sense of this paragraph, the linear scaling is not robust to model inaccuracies. Specifically, it is not $O(1)$-competitive in general, nor even for the case of batch arrivals.

6. FAIRNESS AND SPEED SCALING

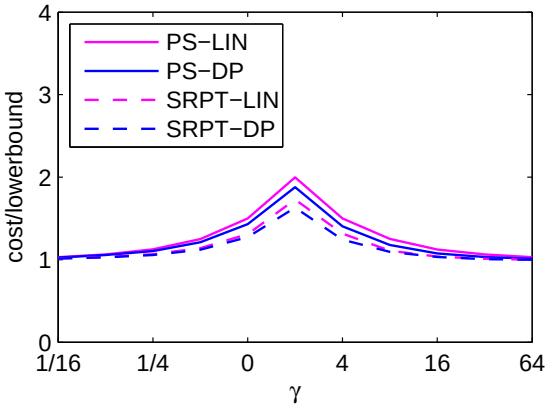
To this point we have seen that speed scaling has many benefits; however we show in this section that dynamic speed scaling has an undesirable consequence — magnifying unfairness. Fairness is an important concern for system design in many applications, and the importance of fairness when considering energy efficiency was recently raised in [29]. However, unfairness under speed scaling designs has not previously been identified. In retrospect though, it is not a surprising byproduct of speed scaling: If there is some job type that is always served when the queue length is long/short it will receive better/worse performance than it would have in a system with a static speed. To see that this magnifies unfairness, rather than being independent of other biases, note that the scheduler has greatest flexibility to select which job to serve when the queue is long, and so jobs served at that time are likely to be those that already get better service.

In this section, we prove that this service rate differential can lead to unfairness in a rigorous sense under SRPT and non-preemptive policies such as First-Come-First-Serve (FCFS, which serves jobs in order of arrival). However, under PS, speed scaling does not lead to unfairness.

[6]This is proven in [33] for $\alpha \in (1, \infty)$.



(a)



(b) c

Figure 7: Comparison of PS and SRPT with linear speeds, $s_n = n\sqrt{\beta}$, and with dynamic speeds optimal for PS. Job sizes are Pareto(2.2) and $P(s) = s^2$.

6.1 Defining fairness

The fairness of scheduling policies has recently received a lot of attention in computer systems modeling, which has led to a variety of fairness measures, e.g., [2, 27, 34], and the analysis of nearly all common scheduling policies, e.g., [20, 26, 34]. Refer to the survey [31] for more details.

Here, we compare fairness not between individual jobs, but between classes of jobs, where a class consists of all jobs of a given size. Since this paper focuses on delay, we compare $\mathbb{E}[T(x)]$ across x. For this purpose, fairness when $s = 1$ has been defined in prior work as follows [31]:

DEFINITION 2. *A policy π is fair if for all x*

$$\frac{\mathbb{E}[T^{\pi}(x)]}{x} \leq \frac{\mathbb{E}[T^{PS}(x)]}{x}.$$

This metric is motivated by the fact that (i) PS is intuitively fair since it shares the server evenly among all jobs at all times; (ii) for $s = 1$, the slowdown ("stretch") of PS is constant, i.e., $\mathbb{E}[T(x)]/x = 1/(1-\rho)$; (iii) $\mathbb{E}[T(x)] = \Theta(x)$ [18], so normalizing by x when comparing the performance of different job sizes is appropriate. Additional support is provided by the fact that $\min_{\pi} \max_{x} \mathbb{E}[T^{\pi}(x)]/x = 1/(1-\rho)$ [34].

Using this definition, it is interesting to note that the class of large jobs is always treated fairly under all work-conserving policies, i.e., $\lim_{x\to\infty} \mathbb{E}[T(x)]/x \leq 1/(1-\rho)$ [18] — even under policies such as SRPT that seem biased against large jobs. In contrast, all non-preemptive policies, e.g., FCFS have been shown to be unfair to small jobs [34].

The foregoing applies when $s = 1$. The following proposition shows that PS still maintains a constant slowdown in the speed scaling environment, and so Definition 2 is still a natural notion of fairness.

PROPOSITION 15. *Consider an M/GI/1 queue with a symmetric scheduling discipline, e.g., PS with controllable service rates. Then, $\mathbb{E}[T(x)] = x\left(\mathbb{E}[T]/\mathbb{E}[X]\right)$.*

PROOF. Since the policy is symmetric, given the number of jobs in the system, the work performed on each job in the system is independent and has p.d.f. $f_e(t) = \bar{F}(t)/\mathbb{E}[X]$.

We apply Little's law to the jobs of size $\in [x, x+\epsilon]$. The arrival rate of such jobs is $\lambda(F(x+\epsilon) - F(x))$, and the expected the number in the system is $\mathbb{E}[N](G(x+\epsilon) - G(x))$ where $G(x)$ $(g(x))$ is the c.d.f. (p.d.f.) of the size of a job in the system. The p.d.f. of the original size X, given $X \geq t$, is $f(X)/\bar{F}(t)$. Then

$$g(x) = \int_0^x \frac{f(x)}{\bar{F}(t)} dF_e(t) = \int_0^x \frac{f(x)}{\mathbb{E}[X]}\, dt = xf(x)/\mathbb{E}[X]$$

where the second step follows from the definition of $F_e(t)$.

By Little's Law, the expected time in system for a job of size $\in [x, x+\epsilon]$, denoted $\mathbb{E}[T([x, x+\epsilon])]$, is

$$\mathbb{E}[T([x, x+\epsilon])] = \left(\frac{\mathbb{E}[N]}{\lambda}\right)\left(\frac{(G(x+\epsilon) - G(x))/\epsilon}{(F(y+\epsilon) - F(y))/\epsilon}\right).$$

Taking the limit as $\epsilon \to 0$ gives the result. □

6.2 Speed scaling magnifies unfairness

Now that we have a natural criterion for fairness, we prove that speed scaling creates/magnifies unfairness under SRPT and non-preemptive policies such as FCFS.

6.2.1 SRPT

We first prove that SRPT treats the largest jobs unfairly in a speed scaling system. Recall that the largest jobs are always treated fairly in the case of a static speed.

Let $\bar{s}^{\pi}$ be the time average speed under policy π, and let $\pi+1$ denote running policy π on a system with a permanent customer in addition to the stochastic load (e.g. $\bar{s}^{PS+1}$).

THEOREM 16. *Consider a GI/GI/1 queue with controllable service rates and unbounded inter-arrival times. Let $s_n^{SRPT} \leq s_n^{PS}$ be weakly monotone increasing and satisfy $\bar{s}^{PS+1} > \rho$ and $\bar{s}^{SRPT+1} > \rho$.*[7] *Then*

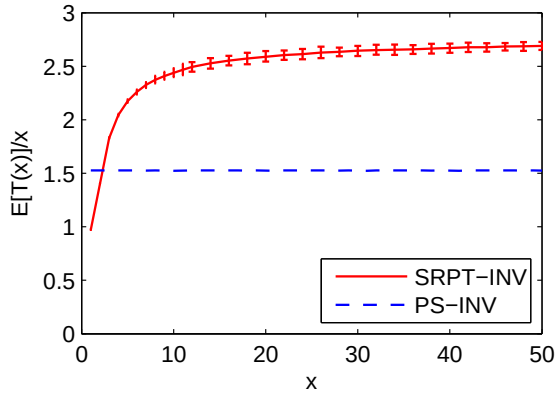
$$\lim_{x\to\infty} \frac{T^{PS}(x)}{x} <_{a.s.} \lim_{x\to\infty} \frac{T^{SRPT}(x)}{x}.$$

The intuition behind Theorem 16 is the following. An infinitely sized job under SRPT will receive almost all of its service while the system is empty of smaller jobs. Thus it receives service during the idle periods of the rest of the system. Further, if $s_n^{SRPT} \leq s_n^{PS}$ then the busy periods will be longer under SRPT and so the slowdown of the largest job will be strictly greater under SRPT. This intuition also provides an outline of the proof.

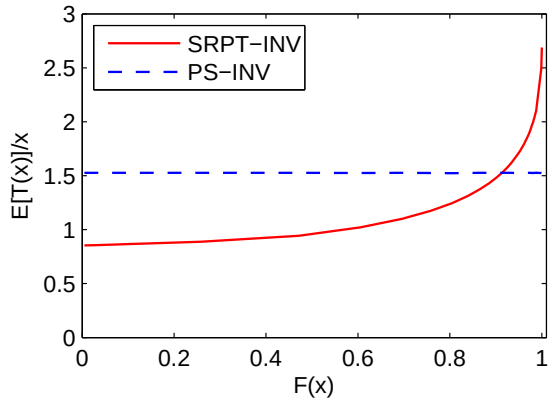
PROOF. By Lemma 21 in Appendix C, $T^{\pi}(x)/x \to 1/(\bar{s}^{\pi+1} - \rho)$ a.s. in each case.

Lemma 24 completes the proof by showing $\bar{s}^{PS+1} > \bar{s}^{SRPT+1}$. It considers the average speed between renewal

[7] Note that the conditions $\bar{s}^{PS+1} > \rho$ and $\bar{s}^{SRPT+1} > \rho$ are equivalent to the stability conditions for s_n^{SRPT} and s_n^{PS}.



(a) vs job size



(b) vs CDF of job size

Figure 8: Slowdown of large jobs under PS and SRPT under Pareto(2.2) job sizes, $\gamma = 1$, $s_n = P^{-1}(n)$, and $P(s) = s^2$. Note the fairness of PS.

instants in which both queues are empty, which it maps to renewal periods. It then uses Lemma 22, which shows that a busy period is longer under SRPT than PS, to show that less work is done on the permanent customer in the renewal period under SRPT than under PS. □

Figure 8 shows that unfairness under SRPT can be considerable, with large jobs suffering a significant increase in slowdown as compared to PS. However, in this case only around 10% of the jobs are worse off than under PS. Note that this setting has a moderate load, which means that SRPT with static speeds would be fair to all job sizes. Figure 8 was generated by running a simulation to steady state and then injecting a job of size x into the system and measuring its response time. This was repeated until the 90% confidence intervals (shown on Figure 8(a) for SRPT) were tight around the estimate.

Theorem 16 proves that SRPT cannot use dynamic speeds and provide fairness to large jobs; however, by using gated-static speed scaling SRPT can provide fairness, e.g., [34]. Further, as Figure 5 illustrates, gated-static speed scaling provides nearly optimal cost. So, it is possible to be fair and near-optimal using SRPT scheduling but, to be fair, robustness must be sacrificed.

6.2.2 *Non-preemptive policies*

The magnification of unfairness by speed scaling also occurs for all non-preemptive policies.

In the static speed setting, all non-preemptive policies are unfair to small jobs [34] since the response time must include at least the residual of the job size distribution if the server is busy, i.e.,

$$\mathbb{E}[T(x)]/x \geq 1 + \rho\mathbb{E}[X^2]/(2\mathbb{E}[X]x),$$

which grows unboundedly as $x \to 0$. However, if we condition on the arrival of a job to an empty system (i.e., the work in system at arrival $W = 0$), then non-preemptive policies are "fair", in the sense that the slowdown is constant: $T(x|W = 0)/x = 1$. Speed scaling magnifies unfairness under non-preemptive policies in the following sense: $T(x|W = 0)/x$ can now differ dramatically across job sizes.

PROPOSITION 17. *Consider a non-preemptive GI/GI/1 speed scaling queue with mean inter-arrival time $1/\lambda$ and speeds s_n monotonically approaching $s_\infty \in (0, \infty]$ as $n \to \infty$. Then, with probability 1,*

$$\lim_{x\to 0} \frac{T(x|W=0)}{x} = \frac{1}{s_1} \quad \textit{and} \quad \lim_{x\to\infty} \frac{T(x|W=0)}{x} = \frac{1}{s_\infty}.$$

The intuition behind this result is that small jobs receive their whole service while alone in the system; whereas large jobs have a large queue build up behind them, and therefore get served at a faster speed. Thus, the service rate of large and small jobs differs, magnifying the unfairness of non-preemptive policies.

PROOF. First, the limit as $x \to 0$ follows immediately from noting that as x shrinks the probability of another arrival before completion goes to 0.

To prove the limit as $x \to \infty$, let $\tilde{A}(x)$ be such that

$$\sum_{i=0}^{\tilde{A}(x)-1} \frac{s_i}{\lambda} < x \leq \sum_{i=0}^{\tilde{A}(x)} \frac{s_i}{\lambda}.$$

and let $\epsilon > 0$ be arbitrary. This $\tilde{A}(x)$ can be thought of as the number of arrivals before x work is completed if jobs arrived periodically with inter-arrival time $1/\lambda$.

Since speeds are non-decreasing, the time to finish the job can be bounded above by the time to reach speed s_i plus the time it would take to finish the whole job at speed s_i. Further, we can use the law of large numbers to bound the time to reach speed s_i as $x \to \infty$. This gives

$$Pr\left(\frac{T(x|W=0)}{x} < \frac{1}{s_{\sqrt{\tilde{A}(x)}}} + \frac{\sqrt{\tilde{A}(x)}}{x}\frac{1+\epsilon}{\lambda}\right) \to 1 \text{ w.p.1.} \tag{28}$$

Since $\{s_i\}$ are non-decreasing and $\tilde{A}(x) = \Theta(x)$, it follows that the right hand side inside the brackets approaches $1/s_\infty$ as $x \to \infty$.

Conversely, a lower bound on the time to finish the job is given by the time to finish it at maximum speed:

$$Pr\left(\frac{T(x|W=0)}{x} \geq \frac{1}{s_\infty}\right) = 1 \text{ w.p.1.} \tag{29}$$

Together, (28) and (29) establish the result. □

In general, speed scaling based on the occupancy n may magnify unfairness in any policy for which $n(t)$ is correlated with the size of the job(s) being processed at time t. Note that gated-static scaling does not magnify unfairness, regardless of the scheduling discipline, since all jobs are processed at the same speed.

7. CONCLUDING REMARKS

This paper has studied several fundamental questions about the design of speed scaling algorithms. The focus has been on understanding the structure of the optimal algorithm, the interaction between speed scaling and scheduling, and the impact of the sophistication of the speed scaler. This has led to several new insights, which are summarized in the introduction.

The analytic approach of this paper is distinctive in that it considers both worst-case and stochastic models. This combination of techniques is fundamental in obtaining two of the main results of the work: Corollary 14 providing worst-case guarantees for policies designed in the stochastic model, and Theorem 16 identifying unfairness in expected performance under dynamic speed scaling with SRPT. Further, the combination of stochastic and worst-case analysis adds support to many of the other insights of the paper, e.g., the decoupling of scheduling and speed scaling.

The results in this paper suggest many interesting topics for future work. Foremost, it will be interesting to see if the lower bound of 2-competitive for natural speed scaling algorithms extends to all algorithms. It is also important to understand the range of applicability of the insights that speed scaling can be decoupled from scheduling with little performance loss, and that scheduling is less important when energy is added to the objective. Further, the study of fairness in the context of speed scaling was only touched on briefly in this paper, and many questions remain. Finally, it is important to address all of the issues studied in this paper in the context of other performance objectives, e.g., when temperature is considered or when more general combinations of energy and response time are considered.

Acknowledgements

This work was supported by NSF CCF 0830511 and CNS 0435520, Microsoft Research, the Lee Center for Advanced Networking and Australian Research Council grant FT0991594. We thank Jeremy Hurwitz for comments on the proof of Theorem 4.

8. REFERENCES

[1] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. In *Lecture Notes in Computer Science (STACS)*, volume 3884, pages 621–633, 2006.

[2] B. Avi-Itzhak, H. Levy, and D. Raz. A resource allocation fairness measure: properties and bounds. *Queueing Systems Theory and Applications*, 56(2):65–71, 2007.

[3] N. Bansal, H.-L. Chan, J. Edmonds, and K. Pruhs. Preprint, 2009. Available ⟨http://www.cs.pitt.edu/∼kirk/postdoc/spaa.pdf⟩.

[4] N. Bansal, H.-L. Chan, T.-W. Lam, and L.-K. Lee. Scheduling for speed bounded processors. In *Proc. Int. Colloq. Automata, Languages and Programming*, pages 409–420, 2008.

[5] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *Proc. SODA*, 2009.

[6] N. Bansal, H.-L. Chan, K. Pruhs, and D. Katz. Improved bounds for speed scaling in devices obeying the cube-root rule. In *Automata, Languages and Programming*, pages 144–155, 2009.

[7] N. Bansal, K. Pruhs, and C. Stein. Speed scaling for weighted flow times. In *Proc. SODA*, pages 805–813, 2007.

[8] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[9] N. Bingham, C. Goldie, and J. Teugels. *Regular Variation*. Cambridge University Press, 1987.

[10] J. R. Bradley. Optimal control of a dual service rate M/M/1 production-inventory model. *European Journal of Operations Research*, 161(3):812–837, 2005.

[11] D. P. Bunde. Power-aware scheduling for makespan and flow. In *Proc. ACM Symp. Parallel Alg. and Arch.*, 2006.

[12] H.-L. Chan, J. Edmonds, T.-W. Lam, L.-K. Lee, A. Marchetti-Spaccamela, and K. Pruhs. Nonclairvoyant speed scaling for flow and energy. In *Proc. STACS*, 2009.

[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[14] T. B. Crabill. Optimal control of a service facility with variable exponential service times and constant arrival rate. *Management Science*, 18(9):560–566, 1972.

[15] J. Edmonds. Scheduling in the dark. In *Proc. ACM STOC*, pages 179–188, 1999.

[16] J. M. George and J. M. Harrison. Dynamic control of a queue with adjustable service rate. *Oper. Res.*, 49(5):720–731, 2001.

[17] S. V. Hanly. Congestion measures in DS-CDMA networks. *IEEE Trans. Commun.*, 47(3):426–437, Mar. 1999.

[18] M. Harchol-Balter, K. Sigman, and A. Wierman. Asymptotic convergence of scheduling policies with respect to slowdown. *Perf. Eval.*, 49(1-4):241–256, Sept. 2002.

[19] F. P. Kelly. *Reversibility and Stochastic Networks*. 1979.

[20] A. A. Kherani and R. Nunez-Queija. TCP as an implementation of age-based scheduling: fairness and performance. In *IEEE INFOCOM*, 2006.

[21] L. Kleinrock. *Queueing Systems Volume II: Computer Applications*. Wiley Interscience, 1976.

[22] T.-W. Lam, L.-K. Lee, I. K. K. To, and P. W. H. Wong. Speed scaling functions for flow time scheduling based on active job count. In *Proc. Euro. Symp. Alg.*, 2009.

[23] M. Lin, A. Wierman, and B. Zwart. Heavy-traffic analysis of mean response time under shortest remaining processing time. Preprint, 2009.

[24] R. Motwani, S. Phillips, and E. Torng. Nonclairvoyant scheduling. *Theoret. Comput. Sci.*, 130(1):17–47, 1994.

[25] K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. In *Scandinavian Worksh. Alg. Theory*, 2004.

[26] I. A. Rai, G. Urvoy-Keller, and E. Biersack. Analysis of FB scheduling for job size distributions with high variance. In *Proc. of ACM Sigmetrics*, 2003.

[27] W. Sandmann. A discrimination frequency based queueing fairness measure with regard to job seniority and service requirement. In *Proc. of Euro NGI Conf. on Next Generation Int. Nets*, 2005.

[28] R. E. Tarjan. Amortized computational complexity. *SIAM J. Alg. Disc. Meth.*, 6(2):306–318, 1985.

[29] P. Tsiaflakis, Y. Yi, M. Chiang, and M. Moonen. Fair greening for DSL broadband access. In *GreenMetrics*, 2009.

[30] R. Weber and S. Stidham. Optimal control of service rates in networks of queues. *Adv. Appl. Prob.*, 19:202–218, 1987.

[31] A. Wierman. Fairness and classifications. *Perf. Eval. Rev.*, 34(4):4–12, 2007.

[32] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *Proc. IEEE INFOCOM*, 2009.

[33] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems: Optimality and robustness. *In preparation*, 2010.

[34] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proc. of ACM Sigmetrics*, 2003.

[35] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 374–382, 1995.

[36] S. Zhang and K. S. Catha. Approximation algorithm for the temperature-aware scheduling problem. In *Proc. IEEE Int. Conf. Comp. Aided Design*, pages 281–288, Nov. 2007.

APPENDIX

A. RUNNING CONDITION FOR SRPT

The proof of Lemma 2 uses the following lemmas, which parallel those in [5]. Let $n^A(\cdot)$ and $n^O(\cdot)$ be arbitrary unfinished work profiles, $n^A = n^A(0)$, $n^O = n^O(0)$, and let s^A and s^O be arbitrary non-negative speeds, with $s^O = 0$ if $n^O = 0$.

LEMMA 18. *For any non-decreasing Δ with $\Delta(i) = 0$ for $i \le 0$, if $n^O < n^A$ then, under SRPT, where Φ is differentiable either*

$$\begin{aligned} \textit{both} \quad & \tfrac{d}{dt}\Phi \le \Delta(n^A - n^O + 1)(-s^A + s^O) && \text{(30a)} \\ \textit{and} \quad & n^O \ge 1, && \text{(30b)} \\ \textit{or} \quad & \tfrac{d}{dt}\Phi \le \Delta(n^A - n^O)(-s^A + s^O). && \text{(30c)} \end{aligned}$$

PROOF. Consider an interval $I = [t, t + dt]$ sufficiently small that no arrivals or departures occur. Let $\Phi(t + dt) - \Phi(t) = d\Phi^A + d\Phi^O$, where $d\Phi^A$ reflects the change in n^A and $d\Phi^O$ reflects the change due to OPT. On I, $n^x[q]$ decreases by 1 for $q \in [q^x - s^x\,dt, q^x]$, for $x = A, OPT$. Then A will remove a term from the sum in (5), and OPT may add an additional term. Let q^A (q^O) be the remaining work of the job being processed by algorithm A (OPT). If $q^A \ne q^O$, these intervals do not overlap, and so

$$d\Phi^A = -\Delta(n^A[q^A] - n^O[q^A])s^A\,dt \quad \text{(31a)}$$

$$d\Phi^O \le \Delta(n^A[q^O] - (n^O[q^O] - 1))s^O\,dt. \quad \text{(31b)}$$

The result follows from one of the following cases, divided by dt. The improvement from [5] comes from handling the boundary case $n^O = 0$ more carefully.

$q^A < q^O$ The second term in (31a) becomes $n^O[q^A] = n^O[q^O] = n^O$, whence $d\Phi^A = -\Delta(n^A - n^O)s^A\,dt$. Since $q^A < q^O$ implies $n^A[q^O] \le n^A[q^A] - 1$, and Δ is non-decreasing, $\Delta(n^A[q^O] - (n^O[q^O] - 1)) \le \Delta(n^A[q^A] - n^O[q^O])$. Thus $d\Phi^A + d\Phi^O \le \Delta(n^A - n^O)(-s^A + s^O)\,dt$.

$q^A = q^O$ If $s^A \ge s^O$ then one term is removed from the sum in (5) for $q \in [q^A - s^A\,dt, q^A - s^O\,dt]$, which gives $\Phi(t + dt) - \Phi(t) = \Delta(n^A - n^O)(-s^A + s^O)\,dt$.

If $s^O > s^A$, then one term is added for $q \in [q^A - s^O\,dt, q^A - s^A\,dt]$, whence $\Phi(t + dt) - \Phi(t) = \Delta(n^A - n^O + 1)(-s^A + s^O)\,dt$. As $s^O > s^A \ge 0$, $n^O \ne 0$, whence $n^O \ge 1$.

$q^A > q^O$ If $n^O = 0$ then, $n^O[q^A] = 0 = n^O$ whence $d\Phi^A \le -\Delta(n^A - n^O)s^A\,dt$, and $s^O = 0$ whence $d\Phi^O = 0 = 2\Delta(n^A - n^O)s^O\,dt$. This implies (30c).

If $n^O > 1$ then $q^A > q^O$ implies $n^O[q^A] \le n^O[q^O] - 1$. Since Δ is non-decreasing, (31a) becomes $d\Phi^A \le -\Delta(n^A - n^O + 1)s^A\,dt$. Since $q^A > q^O$, $n^A[q^O] = n^A[q^A] = n^A$, and (31b) becomes $d\Phi^O \le \Delta(n^A - n^O + 1)s^O\,dt$. This implies (30a) and (30b). □

This differs from the corresponding result in [5] in condition (30b), which ensures that the argument of Δ in (30) is always at most n^A and gives the following.

LEMMA 19. *Consider a regular power function, P, and let $\Delta(i)$ be given by (6) for $i > 0$. If $n^O < n^A$ and $n^A \le P(s^A)/\beta$ then (30) implies*

$$\frac{d\Phi}{dt} \le (1 + \eta)(P(s^O)/\beta - n^A + n^O). \quad \text{(32)}$$

PROOF. Since P is regular, Δ is non-decreasing. Now, consider two cases.

If (30a) and (30b) holds, then let $\Psi(s) = P(s)/\beta$ and set $i = n^A - n^O + 1$ in Lemma 20 below to give

$$\begin{aligned} \frac{d\Phi}{dt} &\le \Delta(i)(-s^A + s^O) = (1 + \eta)\Psi'(\Psi^{-1}(i))(-s^A + s^O) \\ &\le (1 + \eta)\big[(-s^A + \Psi^{-1}(i))\Psi'(\Psi^{-1}(i)) + (\Psi(s^O) - i)\big] \\ &\le (1 + \eta)(\Psi(s^O) - n^A + n^O) \end{aligned}$$

where the last inequality follows from

$$n^O \ge 1 \quad \Rightarrow \quad s^A \ge P^{-1}(n^A\beta) \ge \Psi^{-1}(n^A - n^O + 1). \quad \text{(33)}$$

Otherwise (30c) holds. Since $s^A \ge P^{-1}(n^A\beta) \ge \Psi^{-1}(n^A - n^O)$, the above manipulations go through again, with $i = n^A - n^O$ in Lemma 20. □

Next, we need the following result, which is Lemma 3.1 of [5].

LEMMA 20. *[5] Let Ψ be a strictly increasing, strictly convex, differentiable function. Let $i, s^A, s^O \ge 0$ be any real numbers. Then*

$$\begin{aligned} &\Psi'(\Psi^{-1}(i))(-s^A + s^O) \\ &\quad \le (-s^A + \Psi^{-1}(i))\Psi'(\Psi^{-1}(i)) + \Psi(s^O) - i. \end{aligned}$$

We can now prove Lemma 2.

PROOF OF LEMMA 2. When $n^A = 0$, (7) holds trivially. Consider now three cases when $n^A \ge 1$:

If $n^O > n^A$, then $d\Phi^O = 0$, since there is a $dt > 0$ such that $n^O[q] > n^A[q]$ for $q \in [q^O - s^O\,dt, q^O]$, which implies that $n^O[q] - n^A[q] \le 0$ for all times in $[t, t + dt]$. Since $d\Phi^A \le 0$ on any interval, $d\Phi \le 0$. Thus (7) holds, since $P(s^A)/\beta \le \eta n^A$.

Consider next $n^O < n^A$. Since the optimal scheme runs at zero speed when it is empty, $s^O = 0$ if $n^O = 0$, and so Lemma 18 applies. Then by Lemma 19, $d\Phi/dt \le (1 + \eta)(P(s^O)/\beta - n^A + n^O)$, whence

$$\begin{aligned} n^A + \frac{P(s^A)}{\beta} + \frac{d}{dt}\Phi &\le n^A + \eta n^A + (1 + \eta)(\frac{P(s^O)}{\beta} - n^A + n^O) \\ &= (1 + \eta)(n^O + P(s^O)/\beta). \end{aligned} \quad \text{(34)}$$

Finally, if $n^O = n^A$, then either $d\Phi \le 0$ or (30) holds:

1. If $q^A < q^O$, then $n^A[q] - n^O[q]$ becomes negative for $q \in [q^A - s^A\,dt, q^A]$ (whence $n[q] = \max(0, n^A[q] - n^O[q])$ remains 0), and remains negative for $q \in [q^O - s^O\,dt, q^O]$. Hence $n[q]$ is unchanged, and $d\Phi = 0$.
2. If $q^A = q^O$, consider two cases. (i) If $s^A \ge s^O$ then $n^A[q] - n^O[q]$ becomes negative for $q \in [q^A - s^A\,dt, q^A - s^O\,dt]$ and remains zero for $q \in [q^A - s^O\,dt, q^A]$, whence $n[q]$ again remains unchanged. (ii) Otherwise, $n[q]$ increases by 1 for $q \in [q^A - s^O\,dt, q^A - s^A\,dt]$, and $d\Phi = \Delta(n^A - n^O + 1)(-s^A + s^O)\,dt$. Again, $n^O \ge 1$ since $s^O > s^A \ge 0$, and the optimal is idle when $n^O = 0$.
3. The case $q^A > q^O$ is identical to the case in the proof of Lemma 18, and (30) holds.

Again, if (30) holds, then (7) holds. If instead $d\Phi \le 0$, then the left hand side of (7) is at most $(1 + \eta)n^A$, which is less than the first term on the right hand side. □

B. RUNNING CONDITION FOR PS

PROOF OF LEMMA 6. First note that if $n^A = 0$ then the LHS of (12) is 0, and the inequality holds. Henceforth, consider the case $n^A \ge 1$.

The rate of change of Φ caused by running OPT is at most $\Gamma(n^A)^{1-1/\alpha}s^O$, which occurs when all of the speed is allocated to the job with the largest weight in (11).

Let $l \geq 0$ be the number of zero terms in the sum (11), corresponding to jobs on which PS is leading OPT. The sum in (11) contains $n^A - l$ non-zero terms, each decreasing due to PS at rate $i^{1-1/\alpha}dq^A/dt = i^{1-1/\alpha}s^A/n^A$. The sum is minimized (in magnitude) if these are terms $i = 1, \ldots, n^A - l$. Thus, the change in Φ due to PS is at least as negative as

$$-\Gamma \sum_{i=1}^{n^A-l} i^{1-1/\alpha}\frac{s^A}{n^A} \leq -\Gamma \int_0^{n^A-l} i^{1-1/\alpha}\frac{s^A}{n^A}\,di$$
$$\leq -\Gamma\frac{\alpha}{2\alpha-1}(n^A-l)^{2-1/\alpha}\beta^{1/\alpha}(n^A)^{(1/\alpha)-1} \quad (35)$$

since $s^A \geq (n^A\beta)^{1/\alpha}$. This gives

$$\frac{d\Phi}{dt} \leq \Gamma(n^A)^{1-1/\alpha}s^O - \Gamma\frac{\alpha\beta^{1/\alpha}}{2\alpha-1}(n^A)^{(1/\alpha)-1}(n^A-l)^{2-1/\alpha}$$

Moreover, since $(s^A)^\alpha/\beta \leq \eta n^A$ and $l \leq n^O$, we have $n^A + (s^A)^\alpha/\beta \leq (1+\eta)n^A$ and $n^O + (s^O)^\alpha/\beta \geq l + (s^O)^\alpha/\beta$. To show (12), it is sufficient to show that

$$(1+\eta)n^A + \Gamma(n^A)^{1-1/\alpha}s^O - \Gamma\frac{\alpha\beta^{1/\alpha}(n^A)^{(1/\alpha)-1}(n^A-l)^{2-1/\alpha}}{2\alpha-1}$$
$$\leq c(l + (s^O)^\alpha/\beta).$$

Since $n^A > 0$, dividing by n^A gives the sufficient condition

$$0 \leq c(s^O)^\alpha/(\beta n^A) - \Gamma s^O/(n^A)^{1/\alpha}$$
$$+ cl/n^A + \Gamma\frac{\alpha\beta^{1/\alpha}}{2\alpha-1}(1-l/n^A)^{2-1/\alpha} - (1+\eta). \quad (36)$$

To find a sufficient condition on c, we take the minimum of the right hand side with respect to s^O, l and n^A. Following [3], note that the minimum of the first two terms with respect to s^O occurs for $s^O = (\frac{\beta\Gamma}{c\alpha})^{1/(\alpha-1)}(n^A)^{1/\alpha}$, at which point the first two terms become

$$-\left(1-\frac{1}{\alpha}\right)\left(\frac{\beta\Gamma^\alpha}{c\alpha}\right)^{1/(\alpha-1)}. \quad (37)$$

Now consider a lower bound on the sum of the terms in l. Let $j = l/n^A$, and minimize this with respect to $j \geq 0$. Setting the derivative with respect to j to 0 gives $c = \beta^{1/\alpha}\Gamma(1-j)^{1-1/\alpha}$. Hence the minimum for $j \geq 0$ is for $j = 1 - (\min(1, c/(\beta^{1/\alpha}\Gamma)))^{\alpha/(\alpha-1)}$. For $c \geq \beta^{1/\alpha}\Gamma$, the sum of the terms in l achieves a minimum (with respect to l) of $\beta^{1/\alpha}\Gamma\alpha/(2\alpha-1)$ at $l = 0$, for all n^A. In this case, it is sufficient that

$$0 \leq -\left(1-\frac{1}{\alpha}\right)\left(\frac{\beta\Gamma^\alpha}{c\alpha}\right)^{1/(\alpha-1)} + \beta^{1/\alpha}\Gamma\frac{\alpha}{2\alpha-1} - (1+\eta).$$

Rearranging shows that it is sufficient that $c \geq \beta^{1/\alpha}\Gamma$ and

$$c \geq \beta\left(\frac{\Gamma}{\alpha}\right)^\alpha\left(\frac{(\alpha-1)(2\alpha-1)}{\alpha\beta^{1/\alpha}\Gamma - (1+\eta)(2\alpha-1)}\right)^{\alpha-1}$$
$$= (1+\eta)\left(\frac{2\alpha-1}{\alpha}\right)^\alpha.$$

where the equality uses $\Gamma = (1+\eta)(2\alpha-1)/\beta^{1/\alpha}$. □

C. PROOF OF UNFAIRNESS OF SRPT

The following lemmas establish Theorem 16.

We start by characterizing the limiting slowdown in terms of the average speed in a system with a permanent customer.

LEMMA 21. *Consider a GI/GI/1 queue with service discipline $\pi \in \{PS, SRPT\}$ with controllable service rate s_n^π such that $\bar{s}^{\pi+1} > \rho$, then*

$$\lim_{x\to\infty}\frac{T^\pi(x)}{x} =_{a.s.} \frac{1}{\bar{s}^{\pi+1}-\rho}.$$

PROOF. Consider a PS+1 system. Let $S(t)$ be the total work completed (service given) by time t. Let $R(t)$ be the service given to the permanent job by time t and $\bar{R}(t)$ be the service given to all other jobs by time t. Note $S(t) = R(t) + \bar{R}(t)$.

Since the system is stable, we have $\lim_{t\to\infty}\bar{R}(t)/t =_{a.s.} \rho$, and by definition, we have $\lim_{t\to\infty} S(t)/t =_{a.s.} \bar{s}^{PS+1}$. Thus,

$$\lim_{t\to\infty} R(t)/t =_{a.s.} \bar{s}^{PS+1} - \rho.$$

To complete the proof note that $R(T(x)) = x$ and so

$$\lim_{x\to\infty}\frac{T(x)}{x} = \lim_{t\to\infty}\frac{t}{R(t)} =_{a.s.} \frac{1}{\bar{s}^{PS+1}-\rho}.$$

□

To prove the result in the case of SRPT, we use a parallel argument. We will prove matching upper and lower bounds. First, the upper bound proceeds by considering that $\lim_{t\to\infty}\bar{R}(t)/t \leq_{a.s.} \rho,$. Thus, mimicking the argument for PS, we have

$$\lim_{x\to\infty}\frac{T(x)}{x} = \lim_{t\to\infty}\frac{t}{R(t)} \leq_{a.s.} \frac{1}{\bar{s}^{PS+1}-\rho}.$$

For the corresponding lower bound, consider a truncated job size distribution where jobs of size $> \varepsilon x$ are not considered, for some $\varepsilon > 0$. Then, let $\bar{s}_{\varepsilon x}^{SRPT+1}$ be the average speed given this truncated job size distribution and $\rho_{\varepsilon x}$ be the load of this truncated job size distribution. We have, for large x,

$$\frac{T(x)}{x} \geq \frac{1-\varepsilon}{\bar{s}_{\varepsilon x}^{SRPT+1} - \rho_{\varepsilon x}}$$

by realizing that until the tagged job of size x reaches remaining size εx every arriving job will complete before it. Thus, $\bar{R}(t)/t \to \rho_{\varepsilon x}$. Finally, letting $x \to \infty$ and $\epsilon \to 0$ completes the proof of the lower bound, and the lemma.

Next, we focus on relating the length of the busy periods in the PS and SRPT systems. This eventually lets us conclude that $\bar{s}^{PS+1} > \bar{s}^{SRPT+1}$. Let $t^\pi(w)$ be the time when w work has been completed under policy π.

LEMMA 22. *Consider a single server with a controllable service rate. Assume $s_n^{PS} \geq s_n^{SRPT}$ for all n and that s_n^{PS} and s_n^{SRPT} are both weakly monotonically increasing. Then*

$$t^{PS}(w) \leq t^{SRPT}(w).$$

Hence, busy periods are longer under SRPT than under PS.

PROOF. We prove the result only in the case where $s_n^{PS} = s_n^{SRPT}$ for all n and s_n^{PS} and s_n^{SRPT} are both strictly monotonically increasing; the general proof is analogous.

Observe that $t(w)$ is continuous under both PS and SRPT. So, it is sufficient to show that at every point v when $t^{PS}(v) = t^{SRPT}(v)$ ceases to be true it is because $t^{PS}(v^+) < t^{SRPT}(v^+)$. Thus, we induct over moments when $t^{PS}(v) = t^{SRPT}(v)$ and $t^{PS}(w) \leq t^{SRPT}(w)$ for all $w \leq v$.

The base case follows from noting that $t^{PS}(0) = t^{SRPT}(0) = 0$ and that $s_n^{PS} = s_n^{SRPT}$ until the moment of the first completion, which happens under SRPT due to the optimality of SRPT. Let w_0 be the work that has been completed at this moment. Then, if the system is not empty, $t^{PS}(w_0^+) < t^{SRPT}(w_0^+)$ since s_n is strictly monotonically increasing.

Next, consider a point v such that $t^{PS}(v) = t^{SRPT}(v)$ and $t^{PS}(w) \leq t^{SRPT}(w)$ for all $w \leq v$. There are three cases:

$n^{PS}(t^{PS}(v)) > n^{SRPT}(t^{SRPT}(v))$: In this case, $t^{PS}(v^+) < t^{SRPT}(v^+)$ since s_n is strictly increasing.

$n^{PS}(t^{PS}(v)) = n^{SRPT}(t^{SRPT}(v))$: In this case, $t^{PS}(w) = t^{SRPT}(w)$ for all $w > v$ until the next completion moment, w_0. Applying Lemma 23 below, we know that this completion happens under SRPT. So, $t^{PS}(w_0^+) < t^{SRPT}(w_0^+)$ since s_n is strictly increasing.

$n^{PS}(t^{PS}(v)) < n^{SRPT}(t^{SRPT}(v))$: Lemma 23, below, proves that this cannot happen. □

LEMMA 23. *Consider a single server with a controllable service rate. At moments when $t^{PS}(v) = t^{SRPT}(v)$ and $t^{PS}(w) \leq t^{SRPT}(w)$ for all $w \leq v$,*

$$n^{PS}(t^{PS}(v)) \geq n^{SRPT}(t^{SRPT}(v)).$$

PROOF. The first step is to warp time separately for each system such that the server is always working at rate 1 until v work has been done. To do that, scale time by $1/s_{n(t)}$ at all times t until that point. The warping, and hence the arrival instance, will be different in each system. Call the resulting instances I^{PS} and I^{SRPT}.

These two instances satisfy the following relationships:

(i) The number of arrivals is the same in both instances.

(ii) The size of the ith arrival is the same in both instances for all i.

(iii) The inter-arrival times of the two instances may differ.

(iv) The ith arrival in I^{SRPT} happens no later than the ith arrival in I^{PS}. This follows from the hypothesis of the lemma that $t^{PS}(w) \leq t^{SRPT}(w)$ for all $w \leq v$.

Let $n^{\pi}(t, I)$ denote the number in system at time t in instance I under policy π. Now, it is enough to prove that $n^{PS}(t, I^{PS}) \geq n^{SRPT}(t, I^{SRPT})$. Intuitively, this should be true because (i) the arrivals are happening earlier in I^{SRPT} and (ii) SRPT minimizes the queue length.

To prove this formally, note that the optimality of SRPT immediately gives $n^{PS}(t, I^{PS}) \geq n^{SRPT}(t, I^{PS})$. Second, consider $C^{\pi}(t, I)$, the number of completions by time t under policy π and instance I. To finish the proof, we claim that $C^{SRPT}(t, I^{PS}) \leq C^{SRPT}(t, I^{SRPT})$, whence $n^{SRPT}(t, I^{PS}) \geq n^{SRPT}(t, I^{SRPT})$. To prove the claim, first define a (non–work-conserving) policy Q, which, when run on instance I^{SRPT}, has exactly the same completion instants as SRPT does on instance I^{PS}. Such a Q exists since all arrivals happen no later under I^{SRPT} than I^{PS}.

By the optimality of SRPT and the definition of Q,

$$C^{SRPT}(t, I^{PS}) = C^{Q}(t, I^{SRPT}) \leq C^{SRPT}(t, I^{SRPT}),$$

which completes the proof of the claim and the lemma. □

Finally, we use the above results to prove that $\bar{s}^{SRPT+1} < \bar{s}^{PS+1}$, which completes the proof of Theorem 16.

LEMMA 24. *Consider a GI/GI/1 queue with a controllable service rate and an unbounded inter-arrival time distribution. Assume $s_n^{PS} \geq s_n^{SRPT}$ for all n and s_n^{PS} and s_n^{SRPT} are both weakly monotonically increasing with $s_1 > 0$. Further, assume that $\sup_n s_n^{PS} > \rho$ and $\sup_n s_n^{SRPT} > \rho$. Then both systems are stable and*

$$\rho < \bar{s}^{SRPT+1} < \bar{s}^{PS+1}.$$

PROOF. First, note that the stability of π and $s_1^{\pi} > 0$ guarantees that $\bar{s}_n^{\pi+1} > \rho$ for $\pi \in \{SRPT, PS\}$.

We will prove the result in the case that $s_n^{PS} = s_n^{SRPT}$ for all n. The case of $s_n^{PS} \geq s_n^{SRPT}$ follows immediately. We refer to the PS+1 or the SRPT+1 system as "empty" when it is empty except for the permanent customer. Define a renewal point as occurring when both the PS+1 and SRPT+1 systems are "empty". Since both systems are stable and the inter-arrival time distribution is unbounded, there are infinitely many such renewals.

At the moments when both systems are "empty", the same customers have been completed in both systems. The only difference is how much work has been done on the permanent customer. So, the policy which has completed the most of the permanent customer has the largest average service rate.

We now focus on a single renewal and determine the time average speeds under SRPT+1 and PS+1. By the renewal reward theorem, this is enough to prove the lemma. Let time 0 be the moment at which the sub-busy period began, i.e., the time when both systems were last empty. Let t_e denote the first time when both systems are "empty". Let v denote the amount of work that has been completed on the permanent customer under PS during this sub-busy period.

Now, consider an instance that is unchanged except that the permanent customer is replaced by a job of size z that arrives at time 0 and that no new arrivals occurs after time t_e. Choose z large enough that it will always have lowest priority in the SRPT+1 system. By Lemma 22, this busy period is at least as long under SRPT+1 as under PS+1. Thus, since $s_1^{PS} = s_1^{SRPT}$, at time t_e SRPT+1 must have completed no more than v work on the permanent customer. So, within this renewal, $\bar{s}^{SRPT+1} \leq \bar{s}^{PS+1}$.

Finally, the stronger statement $\bar{s}^{SRPT+1} < \bar{s}^{PS+1}$ holds since with positive probability the renewal will include exactly 2 arrivals that are both in the system at the moment of the first departure under SRPT+1, which guarantees that this completion instant is strictly earlier under SRPT+1 than under PS+1. □