

# Outperforming Multiserver SRPT at All Loads

IZZY GROSOF and DANIELA HURTADO-LANGE, Northwestern University, USA

A well-designed scheduling policy can unlock significant performance improvements with no additional resources. Multiserver SRPT (SRPT- $k$ ) is known to achieve asymptotically optimal mean response time in the heavy traffic limit, as load approaches capacity. No better policy is known for the M/G/ $k$  queue in any regime.

We introduce a new policy, SRPT-Except- $k+1$  & Modified SRPT (SEK-SMOD), which is the first policy to provably achieve lower mean response time than SRPT- $k$ . SEK-SMOD outperforms SRPT- $k$  across all loads and all job size distributions. The key idea behind SEK-SMOD is to prioritize large jobs over small jobs in specific scenarios to improve server utilization, and thereby improve the response time of subsequent jobs in expectation. Our proof is a novel application of hybrid worst-case and stochastic techniques to relative analysis, where we analyze the deviations of our proposed SEK-SMOD policy away from the SRPT- $k$  baseline policy. Furthermore, we design Practical-SEK (a simplified variant of SEK-SMOD) and empirically verify the improvement over SRPT- $k$  via simulation.

## 1 Introduction

The question of reducing wait times with limited resources is ubiquitous in operations and engineering. In particular, strategic scheduling is known to be an effective way to substantially reduce the average time a job spends in the system (and, consequently, the customer waiting time) without incrementing the number of servers. This effectiveness has been observed in various practical applications.

In [3], the authors study delays in cases handled by a courthouse in Italy. Specifically, they design a case-level scheduling policy in which judges estimate the number of hearings needed for each case and schedule all hearings in advance, prioritizing those for cases nearing completion. Compared to the old scheduling policy (where judges scheduled each hearing after completing the previous one), the courthouse reduced average case duration by approximately 12% without increasing resources or the number of judges. Similar scheduling policies have been implemented in multiple courthouses afterward, with comparable results. See, for example, [1, 2].

In computer systems, designing algorithms that minimize response time without adding more resources is essential, as the number of CPUs and GPUs available is typically constant and costly to increase. For example, one can think of a single machine with a fixed number of cores and a limited budget to process tasks. Moreover, this single machine accurately estimates the sizes of arriving tasks. Hence, it can use this information to schedule strategically. For the rest of this paper, we mainly consider this example as an immediate application. Some examples of the importance of scheduling in computer systems with limited resources are the research in [12, 27].

One of the most popular scheduling policies in the literature is First-Come-First-Served (FCFS), where the jobs are processed in order of arrival. This policy is simple to implement and is often considered fair. Due to its simplicity, its analysis is tractable, and has been studied for decades [6, 13, 14, 18]. However, FCFS does not utilize servers strategically: both long and short jobs receive the same priority. Hence, it does not perform well in terms of response time. If one prioritizes shorter jobs over longer ones, the mean response time decreases significantly. Specifically, one can schedule jobs according to the amount of work remaining to complete them, and preemptively process jobs with the Shortest-Remaining-Processing-Time (SRPT) first. For decades, SRPT has been the gold standard for scheduling jobs in single-server queues, as it minimizes the mean response time [20]. However, optimal scheduling for multiple-server systems is much less understood.

---

Authors' Contact Information: Izzy Grosuf, izzy.grosuf@northwestern.edu; Daniela Hurtado-Lange, daniela.hurtado@kellogg.northwestern.edu, Northwestern University, Evanston, Illinois, USA.

A generalization of SRPT to systems with  $k$  servers (called SRPT- $k$ ) is studied in [8], where the  $k$  jobs with the least remaining processing time are scheduled. If there are fewer than  $k$  jobs available, some of the servers are idle. As expected, SRPT- $k$  performs great in terms of the mean response time. However, it has only been proven to minimize this metric in heavy traffic.

A long-lasting open question in the literature is the existence of a scheduling policy that outperforms SRPT- $k$ . In this paper, we answer positively with our policy SRPT-Except- $k+1$  & Modified SRPT (SEK-SMOD). The contributions of this paper are:

- (1) We devise and introduce the SEK-SMOD policy, which is defined in reference to a coupled SRPT- $k$  system, in Section 3.1.
- (2) We prove that SEK-SMOD achieves a smaller mean response time compared to SRPT- $k$  under all loads, all job size distributions, and all numbers of servers  $k \geq 2$  (Theorem 4.1).
- (3) We develop a proof method based on a strategic combination of worst-case scenarios and stochastic analysis, as well as relative analysis. We discuss the proof philosophy in Section 1.2, the proof outline in Section 4.1, and give the proof in Section 5 (stochastic analysis) and Section 6 (worst case analysis).
- (4) We propose the Practical SEK policy (a simplified variant of the SEK-SMOD policy) and empirically demonstrate that it outperforms SRPT- $k$ , via simulation, in Section 7.

### 1.1 Challenges in beating SRPT- $k$

It is well known in the literature that SRPT minimizes the mean response time in single-server queues with known job sizes [20], for any load. Further, the multiserver SRPT- $k$  policy has recently been proven to minimize the mean response time in heavy traffic [8]. SRPT- $k$  also minimizes total response time for all jobs in a no-arrivals (batch) setting. The proof is available in [5], by combining prior results [4, 19]. As a consequence, a scheduling policy that performs better than SRPT- $k$  needs to focus on moderate load.

An important pathway for improving upon SRPT- $k$  is rearranging the order in which jobs are served to improve future parallelism. SRPT- $k$  finishes the smallest jobs first, leaving only a few large jobs. As a result, it reaches states where it cannot fully utilize its servers sooner than other policies. Taking advantage of future parallelism requires a *global* view of the state of the system: Is the system close to empty, with just a few jobs left, or are plenty of jobs present?

In contrast, existing multiserver scheduling analysis has overwhelmingly focused on *index* policies, where each job is evaluated in isolation to produce an index, and the  $k$  jobs of best index are served. Existing analyses for multiserver index policies include FCFS- $k$  [6, 13, 14, 18], class-based priority policies [25], SRPT- $k$  and related index policies [8], M-Gittins- $k$  and M-SRPT- $k$  [23], and Gittins- $k$  [22]. Index policies cannot take a global view of the system, and so cannot look ahead to improve future parallelism. We need a new method of multiserver analysis which can handle non-index scheduling policies.

However, deviating from SRPT- $k$  to improve future parallelism takes on a risk of degrading the mean response time. Intuitively, if jobs arrive in the immediate future, then SRPT- $k$  suffers no loss of parallelism, and any rearrangement is detrimental. On the other hand, if no jobs arrive before the system empties, we are in the batch scenario described above. Only arrivals in the intermediate future can lead to better outcomes for another policy. No prior analysis has attempted to balance these probabilities to ensure a consistent improvement in expectation.

A scheduling policy that performs better than SRPT- $k$  needs to focus on moderate load, needs to examine the global state of the system, and needs to weigh the probabilities of arrivals at different intervals in the future. The SEK-SMOD policy fulfills all of these goals, making possible the first improvement in mean response time over SRPT- $k$ .

## 1.2 Overcoming challenges

We devise the SRPT-Except- $k+1$  & Modified SRPT (SEK-SMOD) scheduling policy, which we prove improves upon the mean response time of the SRPT- $k$  policy, for every job size  $S$ , arrival rate  $\lambda$ , and number of servers  $k \geq 2$ . Our result is formally stated as Theorem 4.1.

SEK-SMOD improves upon SRPT- $k$  by identifying a narrow set of system states where future parallelism can be improved by deviating away from the SRPT- $k$  policy. Each such state involves  $k+1$  jobs present in the system, and in each case SEK-SMOD serves the job of largest remaining size in the system, leaving the second-largest job in the queue. Otherwise, SEK-SMOD serves the same jobs as SRPT- $k$ . These deviations form the SEK part of the combined SEK-SMOD policy.

We prove that each deviation is beneficial, in expectation. The effect on long-term mean response time of this deviation is neutral if no jobs arrive for a long duration after the deviation, and beneficial under the most common arrival patterns, given the Poisson arrival process. The deviation is only harmful under a limited set of scenarios involving arrivals soon after the deviation point, when the predicted improvements in future parallelism fail to materialize. We characterize the frequency and benefit of these deviations through stochastic analysis of the SEK policy in Section 5.

In order to limit the downside of these rare but harmful arrival sequences, the policy switches to the SMOD part of SEK-SMOD. As we show in Section 6, SMOD has a variety of helpful properties relative to a coupled SRPT- $k$  system with the same arrival sequence. For example, the total work present in the SMOD system is always at most as much as the work present in the SRPT- $k$  system, under any arrivals. These worst-case results allow us to bound the consequences of the rare but harmful arrival patterns, and prove that the common arrival patterns always have more impact in expectation. We thereby complete the proof that SEK-SMOD always improves upon SRPT- $k$ .

Our analysis of the SEK-SMOD policy combines two recently developed scheduling analysis frameworks: *hybrid* worst-case and stochastic analysis, and *relative analysis*, where a novel policy is analyzed in terms of its infrequent deviations away from a well-studied baseline policy.

The hybrid analysis framework is most clearly seen in the analysis of the guardrails dispatching policy [7], where jobs are dispatched to servers which each independently serve jobs in SRPT order. Worst-case analysis of the guardrails dispatching policy was combined with stochastic analysis of the SRPT scheduling policy to prove heavy traffic optimality. The hybrid analysis framework is well suited to analyzing non-index policies like the guardrails dispatching policy and the SEK-SMOD policy. Stochastic analysis is best suited to studying well-understood families of scheduling policies (such as index policies), while worst-case analysis often proves weaker results, but can be used in a larger class of policies. Combining both techniques allows strong results to be proven for complex policies, like SEK-SMOD.

The relative analysis framework is most clearly seen in the analysis of the Nudge scheduling policy [10], which deviates slightly from the FCFS scheduling policy to stochastically improve upon its response time distribution. SEK-SMOD similarly deviates slightly from SRPT- $k$  scheduling to improve upon its mean response time. This policy structure allows the baseline and improved policies to be compared on a deviation-by-deviation basis, and allows results demonstrating improvement even when exact analyses for the individual policies are intractable.

Combining the power and generality of hybrid worst-case and stochastic arguments with the deviation-by-deviation framework of relative analysis allows us to prove that SEK-SMOD outperforms SRPT- $k$  for all job size distributions and all arrival rates.

## 2 Prior Work

We discuss prior work on SRPT scheduling in Section 2.1, on hybrid stochastic & worst-case analysis in Section 2.2, on relative analysis in Section 2.3, and on general multiserver scheduling analysis

in Section 2.4. Preliminary work on the design and empirical performance of SEK-SMOD can be found in [9].

## 2.1 Shortest Remaining Processing Time

In a single-server setting, the Shortest Remaining Processing Time (SRPT) is well understood. Its mean response time is exactly characterized in the M/G/1 case [21], and it is known to achieve minimal mean response time in a single-server setting under an arbitrary arrival sequence [20].

In multiserver systems, SRPT was first studied in a batch setting, where all jobs are initially present, and no jobs arrive over time. McNaughton [19] proved that preemption is not useful for minimizing total response time in the batch setting: any preemptive schedule can be matched by a nonpreemptive schedule. Conway et al. [4] proved that the Shortest Processing Time (SPT) policy (the nonpreemptive equivalent of SRPT) is a member of a class of policies that all equally minimize total response time. Graham et al. [5, Section 4.4.1] points out that these results can be combined to prove that SPT (and SRPT) are optimal in the multiserver batch setting with preemption allowed.

A more recent study of multiserver SRPT (also known as SRPT- $k$ ) focused on the heavy-traffic limit, where load approaches capacity for a fixed number of servers. In this limit, SRPT's mean response time is proven to converge to the optimal possible mean response time, that is,  $E[T^{SRPT-k}]/E[T^{OPT}] \rightarrow 1$  as load  $\rho \rightarrow 1$ , for any number of servers  $k \geq 2$  [8]. In the same paper, the Preemptive Shortest Job First (PSJF- $k$ ) and Remaining Size Times Original Size (RS- $k$ ) policies were also proven to converge to optimal in the heavy-traffic limit.

Note that heavy-traffic optimality does not rule out improvement in the heavy-traffic limit, as sub-multiplicative improvement is still possible. Indeed, we prove that SEK-SMOD achieves such improvement over SRPT- $k$  in the heavy traffic limit, as it improves upon SRPT- $k$  at all loads.

## 2.2 Stochastic & Worst-case Hybrid Analysis

A key step in the proof of heavy-traffic optimality for SRPT- $k$  is a worst-case coupling argument, which establishes that the relevant work<sup>1</sup> at some threshold  $x$  present in the SRPT- $k$  system is never more than  $kx$  more than the relevant work in a resource-pooled single-server SRPT system experiencing the same arrivals [8, Lemma 2]. An equivalent result had previously appeared in the worst-case literature, [15, Lemma 2], in the paper where Leonardi and Raz showed that SRPT- $k$  achieves an order-optimal competitive ratio for mean response time with respect to the optimal clairvoyant policy. This worst-case relevant work bound is combined with stochastic tagged-job analysis to upper bound the mean response time of SRPT- $k$ .

A further development of the stochastic-worst-case-hybrid analysis technique is found in the heavy-traffic optimality proof of the guardrails dispatching policy, when combined with SRPT scheduling after jobs are dispatched to their servers [7]. Worst-case proofs bound the difference in relevant work between any two servers at any threshold [7, Lemma 1], as well as between the guardrails system and a resource-pooled single-server system [7, Lemma 2]. These worst-case arguments are again combined with a stochastic tagged-job analysis to upper bound the mean response time of the guardrails dispatching policy.

Our paper builds upon this hybrid framework: our worst-case results in Section 6 compare the SMOD portion of the SEK-SMOD policy against the SRPT- $k$  policy, while past works made worst-case comparisons with much simpler resourced-pooled systems. Our stochastic results in Section 5 also incorporate tagged-job arguments, as well as the relative analysis framework.

---

<sup>1</sup>The relevant work in the system at a threshold  $x$  is the total remaining size of all jobs with remaining size no more than  $x$ .

### 2.3 Relative Analysis

Relative analysis involves comparing two scheduling policies: A baseline policy and a tweaked policy which deviates from that baseline policy in a limited and controlled fashion. The tweaked policy is then analyzed deviation-by-deviation, with the goal of proving that the net expected effects of those deviations improves the performance metric of interest.

This analysis approach was employed to analyze the Nudge scheduling policy for the M/G/1, demonstrating that it achieves stochastic dominance across its response time distribution relative to the FCFS policy, and a multiplicative improvement in its asymptotic tail of response time [10].

This work likewise employs the relative analysis framework, comparing SEK-SMOD to SRPT- $k$  on a divergence-by-divergence level. However, the added complexity of the multiserver setting brings new challenges relative to the M/G/1 setting of the Nudge analysis.

### 2.4 Multiserver Scheduling Analysis: Index Policies

The vast majority of existing multiserver scheduling analysis focuses on *index* policies, where the scheduling policy assigns each job an index based on the job's individual characteristics, and selects the  $k$  jobs with best indices, where  $k$  is the number of servers. Index policies do not use global information, like the total number of jobs in the system, making them easier to analyze.

Analysis of multiserver index policies started with First-Come First-Served (FCFS- $k$ ), where heavy traffic mean response time in the M/G/ $k$  queue was proven to resemble that of a resource-pooled M/G/1 system [6, 13, 14, 18]. Subsequent work has proven similar but much looser bounds for a much broader class of asymptotic environments, beyond heavy traffic [16]. Exact stationary-distribution characterizations have also been proven under exponential service time assumptions, for FCFS- $k$  and two-class priority scheduling [25].

More recently, there has been a family of works on more complex index policies under more general job size distributions, proving that such policies are optimal or near-optimal in the heavy traffic limit, such SRPT- $k$  and related policies [8] in the known-size setting. In the unknown size setting, a variety of policies have been explored and heavy-traffic optimality results proven, such as Foreground-Background (Least Attained Service) [8], Monotonic Gittins and Monotonic Shortest Expected Remaining Processing Time [23], and the Gittins Index policy [11, 22].

All of this prior work has focused on index policies. By contrast, SEK-SMOD is not an index policy. The SEK portion of the policy only diverges from SRPT- $k$  service when exactly  $k + 1$  jobs are present in the system, allowing it to prioritize future parallelism more effectively.

## 3 Model and Definitions

In this paper, we study the known-size M/G/ $k$  queueing system. Jobs arrive according to a Poisson process with rate  $\lambda$ , and have initial state drawn i.i.d. from a general job size distribution with corresponding random variable  $S$ . The system has  $k \geq 2$  servers, each of which serves jobs at rate  $1/k$ : a job of size  $x$  completes after  $kx$  time in service. We focus on the case where the load  $\rho = \lambda E[S] < 1$ , so that stability is possible. We consider a general scheduling policy  $\pi$ , which selects up to  $k$  jobs to serve at any point in time. Service is preempt-resume, so jobs can be paused and returned to the queue with no loss of work. The scheduling policy observes the jobs' size upon arrival and has access to an arbitrary auxiliary Markovian state.

The state of the system at time  $t$  is given by a sorted vector  $\mathbf{r}(t)$  of remaining sizes, where  $0 < r_1(t) \leq r_2(t) \leq \dots \leq r_{n(t)}(t)$  and  $n(t)$  is the number of jobs in the system, possibly combined with an auxiliary state which only affects the scheduling policy. The scheduling policy must be Markovian with respect to this combined state descriptor.

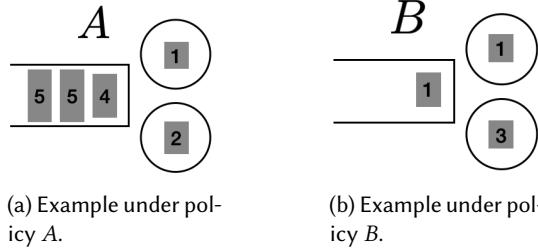


Fig. 1. Example of two M/G/2 systems operating under scheduling policies A and B, respectively. Each gray box represents a job, and the number in each box represents the corresponding remaining size.

Table 1. Example of notation for systems in Fig. 1.

Policy $\pi$	$r^\pi(t)$	$N^\pi(t)$	$d(t)$	$b^\pi(t)$	$W^\pi(t)$
$A$	$r^A(t) = (1, 2, 4, 5, 5)$	$N^A(t) = 5$	$d(t) = 2$	$b^A(t) = (1, 2, 4, 5, 5)$	$W^A(t) = 17$
$B$	$r^B(t) = (1, 1, 3)$	$N^B(t) = 3$	$d(t) = 2$	$b^B(t) = (0, 0, 1, 1, 3)$	$W^B(t) = 5$

We call the system positive recurrent if there exists a fixed state, consisting of no jobs present and a fixed auxiliary state, such that the system returns to this fixed state almost surely and in finite mean time. This paper uses the terms ‘positive recurrence’ and ‘stability’ interchangeably, and focuses on stable systems.

We also consider a coupled system consisting of a pair of M/G/k queues experiencing an identical arrival process under two different scheduling policies. We write the joint state of the coupled system as a pair of sorted vectors of the same length, with padding indices added at the front of the vector of the system with less jobs present.

Specifically, suppose that the individual states of the two queues are the vectors  $\mathbf{r}^A(t)$  and  $\mathbf{r}^B(t)$ , for two policies  $A$  and  $B$ . We define  $N^\pi(t) := |\mathbf{r}^\pi(t)|$  to be the number of jobs in a given system under policy  $\pi \in \{A, B\}$  at time  $t$ . Then, we define the joint state  $\mathbf{bb}(t) := (\mathbf{b}^A(t), \mathbf{b}^B(t))$  as follows, depending on the relation between  $N^A(t)$  and  $N^B(t)$ :

- $N^A(t) = N^B(t)$ : When the two systems have the same number of jobs, we define  $\mathbf{b}^A(t) = \mathbf{r}^A(t)$  and  $\mathbf{b}^B(t) = \mathbf{r}^B(t)$ .
  - $N^A(t) > N^B(t)$ : When  $A$  has more jobs than  $B$ , we define  $\mathbf{b}^A(t) = \mathbf{r}^A(t)$ . Let  $d(t) = N^A(t) - N^B(t)$  be the number of additional jobs in system  $A$  at time  $t$ . We define  $b_i^B(t) = 0$  for all  $i \leq d(t)$ , and define  $b_i^B(t) = r_{i-d(t)}^B(t)$  for all  $i \geq d(t) + 1$ .
  - $N^A(t) < N^B(t)$ : When  $B$  has more jobs than  $A$ , we proceed symmetrically. We define  $\mathbf{b}^B(t) = \mathbf{r}^B(t)$ , let  $d(t) = N^B(t) - N^A(t)$ , and define  $b_i^A(t) = 0$  for all  $i \leq d(t)$ , and  $b_i^A(t) = r_{i-d(t)}^A(t)$  for all  $i \geq d(t) + 1$ .

We define a padding index to be an index  $i$  for which  $b_i^A(t)$  or  $b_i^B(t)$  is zero. The number of padding indices  $d(t) = |N^A(t) - N^B(t)|$ . We use  $W^\pi(t)$  to denote the total remaining work at time  $t$ , in the system under policy  $\pi$ . Specifically,  $W^\pi(t) = \sum_i r_i^\pi(t) = \sum_i b_i^\pi(t)$ .

For example, for the M/G/2 systems under policies  $A$  and  $B$  deployed in Fig. 1, the state descriptors are presented in Table 1.

An important concept in our proofs is a *dominance* relationship between two policies:

*Definition 3.1.* In a coupled system of M/G/k queues experiencing the same arrival process, we define policy  $A$  to *dominate* policy  $B$  whenever  $\forall i, b_i^A \leq b_i^B$ .

Note that we define scheduling policies which have access to the full state of the coupled system. Because all information necessary to simulate the coupled system is known, the state of the coupled system can be taken as the auxiliary Markovian state described above.

For ease of notation, when the time  $t$  is clear from the context, we may omit the explicit reference  $(t)$  and use  $bb$  instead of  $bb(t)$ , for example. In the rest of this paper, we use the notation  $(a)^+ := \max(a, 0)$  for the positive part of a quantity  $a$ .

### 3.1 Defining SEK-SMOD

We now introduce the SRPT-Except- $k + 1$  & Modified SRPT (SEK-SMOD) scheduling policy. This policy is built from two simpler policies, SEK and SMOD, switching between the two based on the counterfactual state of an SRPT- $k$  system, with coupled arrivals but different scheduling.

We start by defining the SEK and SMOD policies individually, and only then define SEK-SMOD. The SEK policy is a standalone policy, defined without reference to any auxiliary state.

*Definition 3.2.* The SRPT-Except- $k + 1$  (SEK) policy is parameterized by constants  $0 \leq \epsilon' < \epsilon \leq x < y$ , as follows.

SEK serves the  $k$  jobs of least remaining size (matching SRPT- $k$ ), except when the following scenario arises:

- (1) There are exactly  $k + 1$  jobs present,
- (2)  $k$  of those jobs have remaining sizes between  $\epsilon'$  and  $\epsilon$ , and
- (3) the largest job has remaining size between  $x$  and  $y$ .

In this case, SEK serves the  $k - 1$  jobs of least remaining size, and the final job of largest remaining size until an arrival or completion occurs.

We now define the SMOD policy, which requires reference to the coupled SRPT- $k$  system.

*Definition 3.3.* The Modified SRPT (SMOD) scheduling policy is defined in relation to a coupled SRPT- $k$  system, that is, based on the combined state  $bb = (b^{SMOD}, b^{SRPT-k})$ .

- If  $N^{SMOD} \leq N^{SRPT-k}$ , SMOD matches SRPT- $k$ .
- If  $N^{SMOD} > N^{SRPT-k}$  and  $N^{SMOD} > k$ , SMOD considers the  $k + d$  jobs with smallest remaining size *eligible for service* (recall  $d := N^{SMOD} - N^{SRPT-k}$ ). SMOD gives *high priority* to eligible jobs such that  $b_i^{SMOD} - b_i^{SRPT-k} \geq 0$ , that is, if the SMOD job's remaining size is larger than or equal to the SRPT- $k$  job. We call these indices *zero-or-positive difference* indices.
  - If there are at least  $k$  zero-or-positive-difference eligible indices, SMOD serves the  $k$  jobs with least remaining size among them.
  - Otherwise, SMOD serves all the zero-or-positive-difference eligible indices, and fills the remaining servers with the least-remaining-size jobs among the negative-difference eligible indices.

Note that all of the padding indices  $i \leq d$  are zero-or-positive-difference indices, and some of the remaining  $k$  eligible indices,  $d < i \leq k + d$ , may be zero-or-positive-difference as well.

As shown in Section 7, the Practical SEK algorithm (Definition 7.1) exhibits excellent performance in simulations. The Practical SEK policy depends on a reduced number of parameters, simplifying the definitions above. However, it is difficult to prove that it always improves on SRPT- $k$ , as there are unlikely scenarios where not following SRPT- $k$  may incur a high cost. To control for this, we define the SEK-SMOD policy, which “protects” the system’s performance by only using SEK when it is provably beneficial.

SEK-SMOD is defined in terms of *divergence points*, which we define to be each moment in time that begins an interval during which the SEK-SMOD policy differs from SRPT- $k$ . Further, we refer to any state in which a divergence point can occur as *divergence starting state*, that is, a state where SEK service is in use, and where the system state satisfies the SEK divergence conditions given in Definition 3.2.

We are now ready to define the combined SEK-SMOD scheduling policy.

*Definition 3.4.* The SRPT-Except- $k+1$  & Modified SRPT (SEK-SMOD) scheduling policy is defined in reference to a coupled SRPT- $k$  system with the same arrivals, and parameterized by constants  $0 \leq \epsilon' < \epsilon \leq x < y$ , as follows.

- If the SRPT- $k$  and SEK-SMOD systems have identical states, the SEK-SMOD policy serves jobs according to the SEK policy given in Definition 3.2.
- If the systems do not have identical states, and no jobs have arrived since the most recent divergence point, then SEK-SMOD continues according to the SEK policy.
- If the systems do not have identical states, and there has been an arrival since the most recent divergence point, then SEK-SMOD uses the SMOD scheduling policy from Definition 3.3.

We focus on the following parameterization of SEK-SMOD to prove our main result, Theorem 4.1. Under this parameterization, we show that SEK-SMOD always improves upon SRPT- $k$ .

*Definition 3.5.* For a given  $\lambda, S$ , and  $k \geq 2$  setting, we define following SEK-SMOD parameters. We start by defining  $x$  to be an arbitrary size such that  $P(S \in [x, 2x]) > 0$ , and  $y = 2x$ .

To define  $\epsilon$ , we first define the constants  $c_1, c_2, c_3, c_4$ , which are used in stating Lemmas 5.2 to 5.4:

$$\begin{aligned} c_1 &= 2k\lambda, & c_2 &= k \left( \lambda \frac{(k+1)y + kx/6}{1 - \rho_{\leq y}} + k + 2 \right), \\ c_3 &= \frac{(\lambda kx/3)^k e^{-\lambda k(y+8x/3)}}{k!} \mathbb{P}(S \in [x, 2x])^k, & c_4 &= k/2, \end{aligned}$$

where  $\rho_{\leq y} := \lambda \mathbb{E}[S \mathbf{1}\{S \leq y\}]$  is the load of jobs with size at most  $y$ . With these constants, we define  $\epsilon = \min(\frac{x}{6}, \frac{c_3 c_4}{2c_1 c_2})$  and  $\epsilon' = \frac{\epsilon}{2}$ , completing the parameterization.

Note that for simplicity, we specify an exact parametrization above. However, our proof has more flexibility:  $\epsilon$  may be any sufficiently small value above 0, and  $y$  may be any sufficiently large value above  $x$ . These parameters are sufficient for our proof, and do not optimize them in this paper.

### 3.2 Defining performance measures

A job's response time is the duration from its arrival to its completion. We define  $T^\pi$  to be a random variable denoting a generic job's response time in stationarity under the scheduling policy  $\pi$ . This paper focuses on analyzing the mean response time  $\mathbb{E}[T^\pi]$  for scheduling policies  $\pi$ .

To analyze mean response time  $\mathbb{E}[T^\pi]$ , we make use of Little's law,  $\lambda \mathbb{E}[T^\pi] = \mathbb{E}[N^\pi]$ . Specifically, we show that  $\mathbb{E}[T^{\text{SEK-SMOD}}] < \mathbb{E}[T^{\text{SRPT-}k}]$ , or equivalently that  $\mathbb{E}[N^{\text{SEK-SMOD}}] < \mathbb{E}[N^{\text{SRPT-}k}]$ . To prove this, we analyze the integrated difference in number of jobs between the two systems, starting at an SEK divergence point, and ending when the SEK-SMOD and SRPT- $k$  systems are next identical.

*Definition 3.6.* The *integrated number difference* (IND) starting at a given SEK divergence point, which occurs at time  $t$ , is defined as follows.

Let  $\tau$  be the next time after  $t$  when the SRPT- $k$  and SEK-SMOD systems have identical states and define  $\Delta_t$  as the difference in number of jobs between the SRPT- $k$  and SEK-SMOD systems,

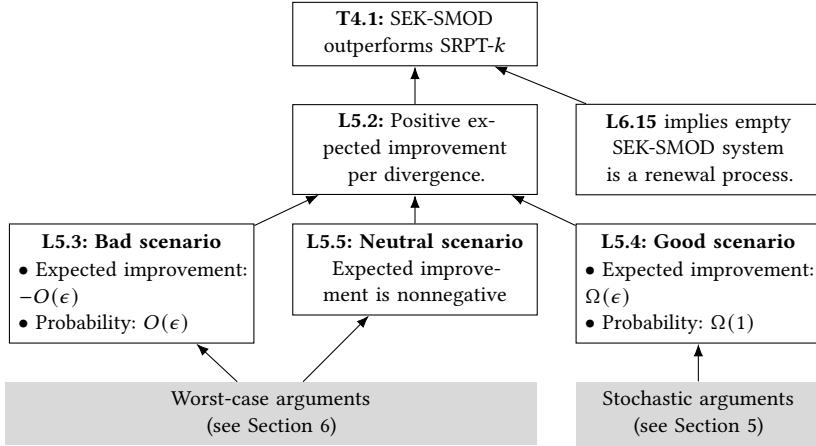


Fig. 2. Diagram of proof of Theorem 4.1.

integrated over the interval  $[t, \tau]$ . Explicitly, we define  $\Delta_t$ :

$$\Delta_t := \int_{u=t}^{\tau} N^{SRPT-k}(u) - N^{SEK-SMOD}(u) du.$$

#### 4 Main result

We show that the SRPT-Except- $k+1$  & SRPT Modified (SEK-SMOD) policy, defined in Definition 3.4, achieves lower mean response time in the M/G/k system than the previously-state-of-the-art SRPT- $k$  policy, for all job sizes  $S$ , all arrival rates  $\lambda$ , and all numbers of servers  $k \geq 2$ . Intuitively, the SEK-SMOD policy identifies a family of situations where SRPT- $k$  performs suboptimally, and deviates away from SRPT- $k$  only in those situations.

These deviations occur when there is a tradeoff between immediately serving jobs with low remaining size, and saving jobs to increase utilization in the future. SRPT- $k$  always prioritizes the smallest jobs, while SEK-SMOD selectively prioritizes future utilization. We demonstrate that the balance of probabilities favors prioritizing future utilization in the situations we identify, allowing SEK-SMOD to outperform SRPT- $k$ .

Our main result is as follows:

**THEOREM 4.1.** *For all arrival rates  $\lambda$ , job sizes  $S$ , and numbers of servers  $k \geq 2$ , with the SEK-SMOD parameterization given in Definition 3.5, if M/G/k/SRPT is positive recurrent, the mean response time of SEK-SMOD is smaller than that of SRPT- $k$ :*

$$\mathbb{E}[T^{SEK-SMOD}] < \mathbb{E}[T^{SRPT-k}].$$

PROOF DEFERRED TO SECTION 5.1. □

Before providing the detailed proof, we provide an outline in Section 4.1. The proof is broadly split into stochastic and worst-case arguments. The stochastic arguments focus on the SEK portion of the policy, and the corresponding lemmas are proved in Section 5. The worst-case proofs focus on the SMOD portion of the policy, and are presented in Section 6.

#### 4.1 Proof outline

We discuss the key ideas in our proof, which are also summarized in Fig. 2.

To prove that the SEK-SMOD policy always improves upon the SRPT- $k$  policy, we take a divergence-by-divergence approach. This is possible because the M/G/ $k$ /SEK-SMOD system is positive recurrent, and we show in Section 5.1 that the empty system defines a renewal process.

Note that a positive expected IND is equivalent to mean response time improvement, as we show in Section 5.1. We therefore focus on characterizing the effects of each SEK divergence on the expected IND. Specifically, we show that the expected IND is positive on each SEK divergence from SRPT- $k$  (Lemma 5.2).

Once SEK diverges from SRPT- $k$ , an arrival happening too soon may result in negative IND. However, this happens with a small probability and is compensated by the gains when there are no arrivals for a relatively short period of time. Lemma 5.2 formalizes this intuition by distinguishing three scenarios. For each of them, we bound the expected IND and frequency to obtain our main result. We provide an intuition to these scenarios below, and a formal definition in Section 5:

- A *good scenario* is when no jobs arrive for a substantial period of time after a divergence point. Then, the schedule selected by SEK-SMOD benefits the IND for sufficiently long, and provides an improvement of  $\Omega(\epsilon)$ . Because arrivals occur according to a Poisson process, this event happens with a positive probability, lower bounded by a constant. See Lemma 5.4.
- A *bad scenario*, on the contrary, happens when a job arrives immediately after a divergence point. This situation may yield a decrease in IND by  $O(\epsilon)$ , but it is rare and, indeed, happens with probability  $O(\epsilon)$ . Hence, a bad scenario contributes with a negative effect of at most  $O(\epsilon^2)$ . See Lemma 5.3 for a formalization of this result and the corresponding proof.
- All other possibilities are called a *neutral scenario*, and we show that they have a nonnegative effect on the expected IND. See Lemma 5.5.

We specifically study these scenarios as  $\epsilon \rightarrow 0$ , limiting the size of the  $k$  smallest jobs at the divergence point to smaller and smaller sizes. We demonstrate that the advantage of SEK-SMOD policy over the SRPT- $k$  policy is clearest as this  $\epsilon$  gets smaller, and specifically, that there exists an explicit  $\epsilon$  threshold small enough such that the expected IND is always positive, given in Definition 3.5.

The stochastic results in the three scenarios are underpinned by worst-case results for the SMOD policy, especially in the bad scenario. We provide a discussion and a formal proof of these worst-case results in Section 6.

Our worst-case results allow us to shift our attention from the duration until the SEK-SMOD system is *identical* to the SRPT- $k$  system, to the duration until the SEK-SMOD system *dominates* SRPT- $k$  system (see Definition 3.1). Intuitively, when SEK-SMOD dominates SRPT- $k$ , the SEK-SMOD system state is strictly preferable to that of the SRPT- $k$  system. We prove in Corollary 6.3 that this property is maintained until the next divergence point. We then bound the time until dominance in Lemma 6.14, and in Lemma 6.7 we bound the IND in terms of the number jobs that arrive before dominance. These combine to give the IND bounds that we need.

These worst-case bounds are incorporated into our stochastic results for the good, bad, and neutral scenarios, demonstrating the desired behavior in the  $\epsilon \rightarrow 0$  limit. This behavior implies positive expected IND for any  $\epsilon$  smaller than an explicit threshold, which in turn implies our main result, Theorem 4.1: SEK-SMOD with the parameterization given in Definition 3.5 always improves on SRPT- $k$ .

## 5 Stochastic proofs

In this section, we prove our main result, Theorem 4.1. To do so, we prove that each SEK-SMOD divergence has a positive expected IND. The timing of the next arrival after a divergence point and the corresponding job size have a significant influence on the number of jobs present in each system

at each time interval, determining the value of the IND. In Lemma 5.2, we identify three possible scenarios and study the corresponding IND separately. We start by defining these scenarios:

**Definition 5.1.** Define three events in the time immediately following a divergence point:

**Bad scenario:** A job arrives during the next  $2k\epsilon$  time after the divergence point (i.e. before all small jobs are guaranteed to finish under SEK-SMOD).

**Good scenario:** No job arrives during the next  $kx/3 > 2k\epsilon$  time after the divergence point, and then  $k$  big enough jobs arrive. These are large enough to wait in line until the largest job in service completes. Specifically, letting the size of the largest job at time of divergence be  $b$ , we require that exactly  $k$  jobs with sizes in  $[x, 2x]$  arrive during the window  $[k(b - 2x/3), k(b - x/3)]$ , and no other jobs arrive before all  $k$  of those jobs are done and the originally-largest job is done, at time  $k(b + 2x)$ .

**Neutral scenario:** Everything else.

We define the events  $\text{BAD}_t$ ,  $\text{GOOD}_t$ , and  $\text{NEUTRAL}_t$  to occur when the corresponding scenario occurs due to a divergence at time  $t$ .

In Lemmas 5.3 to 5.5, respectively, we lower bound the IND in each scenario, based on the stochastic properties of the SEK policy. These stochastic results are underpinned by worst-case results on the SMOD policy, which we prove in Section 6. Finally, in Section 5.1, we bring these results together to prove our main result, Theorem 4.1.

**LEMMA 5.2.** *For any  $\lambda, S, k$ , under the SEK-SMOD parameterization in Definition 3.5, and for any divergence starting state  $\mathbf{bb}$ , the integrated number difference is positive. Specifically,*

$$\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb}] \geq \epsilon c_3 c_4 - \epsilon^2 c_1 c_2,$$

which is positive whenever the parameter  $\epsilon < \frac{c_3 c_4}{c_1 c_2}$ .

Note that Lemma 5.2 provides a lower bound on the performance improvement of SEK-SMOD compared to SRPT- $k$ . In Section 7 we show that the practical improvement may be considerably higher, even for a simplified version of the SEK-SMOD policy.

**PROOF.** To compute a lower bound on  $\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb}]$ , we condition on the possible scenarios. We show the following lower bounds for any divergence starting state  $\mathbf{bb}$ , where  $c_1, c_2, c_3, c_4$  are defined in Definition 3.5. We focus on proving lower bounds that are tight in the  $\epsilon \rightarrow 0$  limit, for any divergence starting state  $\mathbf{bb}$ . Specifically, we show:

**Bad:** (Lemma 5.3)  $P(\text{BAD}_t \mid \mathbf{bb}(t) = \mathbf{bb}) \leq c_1 \epsilon$ , and  $E[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{BAD}_t] \geq -c_2 \epsilon$ .

**Good:** (Lemma 5.4)  $P(\text{GOOD}_t \mid \mathbf{bb}(t) = \mathbf{bb}) \geq c_3$ , and  $E[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{GOOD}_t] \geq c_4 \epsilon$ .

**Neutral:** (Lemma 5.5)  $E[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{NEUTRAL}_t] \geq 0$ .

Recall from Definition 3.5 that the constants  $c_1, c_2, c_3, c_4$  depend on  $\lambda, S, k$ , but not on  $\epsilon$ .

Then, by the law of total probability and using the bounds listed above, we see that

$$\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb}] \geq -\epsilon^2 c_1 c_2 + \epsilon c_3 c_4 + 0$$

With these lower bounds, we see that for any  $\epsilon < \frac{c_3 c_4}{c_1 c_2}$ , as in the parameterization in Definition 3.5, we have  $E[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb}] > 0$  for any divergence starting state  $\mathbf{bb}$ , as desired.  $\square$

We now formalize the results to lower bound the IND for the bad, good, and neutral scenarios:

**LEMMA 5.3 (BAD SCENARIO).** *With the constants  $c_1, c_2 > 0$  defined in Definition 3.5 that depend on  $\lambda, S, k$  and not on  $\epsilon$ , such that for any SEK-SMOD parameterization as in Lemma 5.2 and any*

divergence starting state  $\mathbf{bb}$ ,

$$P(BAD_t \mid \mathbf{bb}(t) = \mathbf{bb}) \leq c_1\epsilon, \quad (1)$$

$$\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& BAD_t] \geq -c_2\epsilon. \quad (2)$$

**PROOF.** Let's start with (1). Note that for the bad event to happen, a job must arrive in the next  $2k\epsilon$  time. This occurs with probability  $1 - e^{-2k\lambda\epsilon}$ , as the arrival process is a Poisson process with rate  $\lambda$ . Note that  $1 - e^{-2k\lambda\epsilon} \leq 2k\lambda\epsilon$ . Thus, because  $c_1 = 2k\lambda$ , we confirm (1).

Next, let's turn to (2). To prove this, we invoke the worst-case properties of SEK-SMOD proven in Section 6. Lemma 6.7 bounds the first time after  $t$  when the SEK-SMOD system dominates the SRPT- $k$  system, which we call  $t_{dom}$ . Lemma 6.14 gives us a bound on  $\mathbb{E}[t_{dom} \mid \mathbf{bb}(t) = \mathbf{bb} \& BAD_t]$  that depends on the completion time of the largest job present at the time of divergence. By Lemma 6.15, both properties apply to the SEK-SMOD policy, not just the SMOD policy. The completion time of the largest job present at the time of divergence can be upper bounded using multiserver tagged job analysis of [8], which completes the proof. We provide the details in Appendix B.  $\square$

**LEMMA 5.4 (GOOD SCENARIO).** *With the constants  $c_3, c_4 > 0$  defined in Definition 3.5 that depend on  $\lambda, S, k$  and not on  $\epsilon$ , and any divergence starting state  $\mathbf{bb}$ ,*

$$P(GOOD_t \mid \mathbf{bb}(t) = \mathbf{bb}) \geq c_3, \quad \mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& GOOD_t] \geq c_4\epsilon.$$

**PROOF.** First, to characterize  $P(GOOD_t \mid \mathbf{bb}(t) = \mathbf{bb})$ , note that the good scenario described in Definition 5.1 does not depend on  $\epsilon$  at all. We compute its probability and verify it's at least  $c_3$  in Appendix C.1.

We now bound  $\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& GOOD_t]$ . Let us compare the behavior under SRPT- $k$  with that under SEK-SMOD. The rest of the proof can be summarized in the following situations, depending on which jobs are being counted in the IND. Each of them has an effect  $\Delta_t$ , characterized below.

- When counting the  $k - 1$  jobs with least remaining size at time of divergence, both systems have the same completion time.
- When counting the second-largest job at time of divergence, SRPT- $k$  completes earlier by  $kb_1$ , where  $b_1$  is the remaining size of the smallest job at time of divergence, under  $\mathbf{bb}$ .
- When counting largest job at the time of divergence, SEK-SMOD completes earlier by  $kb_1$ .
- If there are later arriving jobs before merging, then  $k - 1$  of the jobs have the same completion times, and one has a completion time  $kb_1$  earlier in the SEK-SMOD system.

Note that  $\Delta_t$  is equivalent to the difference in total response time over all jobs that arrive after the divergence at time  $t$  and prior to merging. Consequently, we show that

$$\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& GOOD_t] = kb_1 \geq k\epsilon/2, \quad c_4 = k/2.$$

The proof of each bullet point is simple, as it only requires careful analysis of the remaining job sizes. We provide the details in Appendix C.2.  $\square$

**LEMMA 5.5 (NEUTRAL SCENARIO).** *For the SEK-SMOD parameterization given in Definition 3.5 and any allowed starting state  $\mathbf{bb}$ , all neutral and good scenarios achieve nonnegative  $\Delta_t$ :*

$$[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& (GOOD_t \wedge NEUTRAL_t)] \geq 0, \text{ always.}$$

**PROOF.** In the good and neutral scenarios, no job arrives in the next  $2k\epsilon$  time after the divergence point, which occurs at time  $t$ . The integrated difference from time  $t$  to  $t + 2k\epsilon$  is  $-kb_1$ , which equals the remaining size of the smallest job in  $\mathbf{bb}$  at time of divergence:

$$\int_{u=t}^{t+2k\epsilon} N^{SRPT-k}(u) - N^{SEK-SMOD}(u) = -kb_1. \quad (3)$$

To show this, we can examine the completion times or lack thereof of each of the  $k + 1$  jobs which are present at divergence under  $\mathbf{bb}$ , up to time  $t + 2k\epsilon$ .

To complete the proof, we analyze the IND under three possible situations:

- The  $k - 1$  jobs of smallest remaining size all are in service at the divergence time under both policies, and finish at the same time under both policies. This has no effect on the integrated difference.
- The job of second-largest remaining size is in service at time divergence time  $t$  under the SRPT- $k$  policy, and finishes at time  $t + kb_k$ . Under the SEK-SMOD policy, the same job only enters service at time  $t + kb_1$ , when the job of smallest remaining size completes, and finishes at time  $t + k(b_1 + b_k)$ . Note that  $b_1 \leq b_k \leq \epsilon$ , so  $k(b_1 + b_k) \leq 2k\epsilon$ , so this completion occurs in the desired time window. This generates the  $-kb_1$  difference described above.
- The job of largest remaining size has not finished by time  $2k\epsilon$  under either policy, as  $b_{k+1} \geq x > 6\epsilon$ . This has no effect on the integrated difference.

At time  $t + 2k\epsilon$ , the state of the SRPT- $k$  system is a single job with remaining size  $b_{k+1} - (2\epsilon - b_1)$ , as the largest job entered service at time  $t + kb_1$ , and has received  $(2\epsilon - b_1)$  service. The state of the SEK-SMOD system is a single job with remaining size  $b_{k+1} - 2\epsilon$ , as the largest job entered service at time  $t$  and has received  $2\epsilon$  service. In each system, all other jobs have completed and no jobs have arrived.

Note in particular that SEK-SMOD dominates SRPT- $k$  at this point in time, because the SEK-SMOD job has less remaining size. We now invoke Lemma 6.9, which shows that the IND over the interval from  $t + 2k\epsilon$  to the time of the next merge  $\tau$ ,

$$\int_{u=t+2k\epsilon}^{\tau} N^{SRPT-k}(u) - N^{SEK-SMOD}(u) \geq kb_1. \quad (4)$$

Combining the intervals in (3) and (4), we find that  $\Delta_t \geq 0$ , always.  $\square$

## 5.1 Proof of Theorem 4.1

We are now ready to prove Theorem 4.1, our main result.

**THEOREM 4.1.** *For all arrival rates  $\lambda$ , job sizes  $S$ , and numbers of servers  $k \geq 2$ , with the SEK-SMOD parameterization given in Definition 3.5, if M/G/k/SRPT is positive recurrent, the mean response time of SEK-SMOD is smaller than that of SRPT- $k$ :*

$$\mathbb{E}[T^{SEK-SMOD}] < \mathbb{E}[T^{SRPT-k}].$$

**PROOF.** By Little's Law [17], the claim is equivalent to proving that

$$\mathbb{E}[N^{SEK-SMOD}] < \mathbb{E}[N^{SRPT-k}] \iff \mathbb{E}[N^{SRPT-k} - N^{SEK-SMOD}] > 0.$$

We focus on this version of the claim.

We analyze the behavior of the joint (SEK-SMOD, SRPT- $k$ ) system as a series of cycles. First, SEK-SMOD diverges from SRPT- $k$ . Next, the two systems eventually *merge*, that is, they have the same state once more. Finally, another cycle begins with another divergence. Note that the difference  $N^{SRPT-k} - N^{SEK-SMOD}$  can only be nonzero when the system states are different.

To analyze the behavior of the joint system, we use a renewal-reward argument. To do so, we need a renewal point that has finite expected time between renewals. Our renewal points are the times when both systems are empty, and a job arrives. We know that the SRPT- $k$  system has finite mean time between visits to the empty state. We need to show that there is a finite mean time between visits to the jointly empty state.

We prove this by reference to our worst case lemmas. Lemma 6.15, building on Lemma 6.12, shows that the work under SEK-SMOD never exceeds work under SRPT- $k$ :  $W^{SEK-SMOD} \leq W^{SRPT-k}$ .

Therefore, an empty SRPT- $k$  system ( $W^{SRPT-k} = 0$ ) implies  $W^{SEK-SMOD} = 0$ , and thus the SEK-SMOD system is empty as well. Hence, the jointly empty state is a valid renewal point with finite expected time per renewal.

We apply the renewal reward formula, where our reward function is  $N^{SRPT-k} - N^{SEK-SMOD}$ :

$$\mathbb{E}[N^{SRPT-k} - N^{SEK-SMOD}] = \frac{\mathbb{E}\left[\int_{u \in \text{cycle}} N^{SRPT-k}(u) - N^{SEK-SMOD}(u) du\right]}{\mathbb{E}[\text{Length of cycle}]}.$$

Our goal is to show that the numerator, the expected reward per both-empty cycle, is positive. We can quantify this total reward per both-empty cycle in terms of the expected total reward per divergence event in a given cycle, that is,

$$\mathbb{E}\left[\int_{u \in \text{cycle}} N^{SRPT-k}(u) - N^{SEK-SMOD}(u) du\right] = \mathbb{E}\left[\sum_{t \in \text{cycle divergence points}} \Delta_t\right].$$

Note that this sum is valid because  $\tau$ , the merging time at which  $\Delta_t$  completes, is bounded by the time at which the jointly empty state next occurs. We show that this expected sum is positive.

Lemma 5.2 proves that  $\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb}] > 0$ , for any arbitrary divergence starting state  $\mathbf{bb}$ . Observe that the number of divergences per cycle is irrelevant because each divergence contributes a positive expectation, and there is a positive expected number of divergences per cycle. Additionally, because we condition on an arbitrary starting state, correlations between different divergences are also irrelevant to the analysis.

This result implies that the expected total reward per both-empty-cycle is positive, because the probability of a divergence event is positive. Therefore,  $E[T^{SEK-SMOD}] < E[T^{SRPT-k}]$  as desired.  $\square$

## 6 Worst case proofs

In this section, we prove key worst-case properties of the SMOD policy and the SEK-SMOD policy, which are the foundational results that underpin our stochastic results in Section 5. Specifically, in Section 6.1, we characterize the worst-case behavior of the SMOD policy relative to an SRPT- $k$  system experiencing the same arrival sequence. Then, in Section 6.2, we specialize these results to characterize the combined SEK-SMOD policy, again relative to the SRPT- $k$  policy.

### 6.1 SMOD Worst case lemmas

In this section, we analyze the behavior of the joint system containing an SMOD queue and an SRPT- $k$  queue under an arbitrary stream of arrivals. Specifically, we prove a series of monotonicity and preservation lemmas, showing that the SMOD policy monotonically improves (or preserves) relative to the SRPT- $k$  system, under arbitrary arrivals. We also prove properties of the IND under these worst-case arrivals, which are defined in the same section as the corresponding lemma.

We provide a diagram representing how these lemmas fit in the proof of Theorem 4.1 in Fig. 3. The diagram refers to new concepts that are key to our proof and are defined later in this section. Specifically, we refer to the positive part (Definition 6.1), zig-zag matching (Definition 6.4), and PLN (Definition 6.11). When we mention the “work inequality” property, we mean that  $W^A(t) \leq W^{SRPT-k}$ , where policy  $A$  can be SMOD or SEK-SMOD depending on the context.

**6.1.1 Positive part of work difference:  $r^+$ .** Intuitively, we show that the margin by which SMOD is in a worse state than SRPT- $k$  never increases. As a corollary, dominance of SMOD over SRPT- $k$  is preserved into the future.

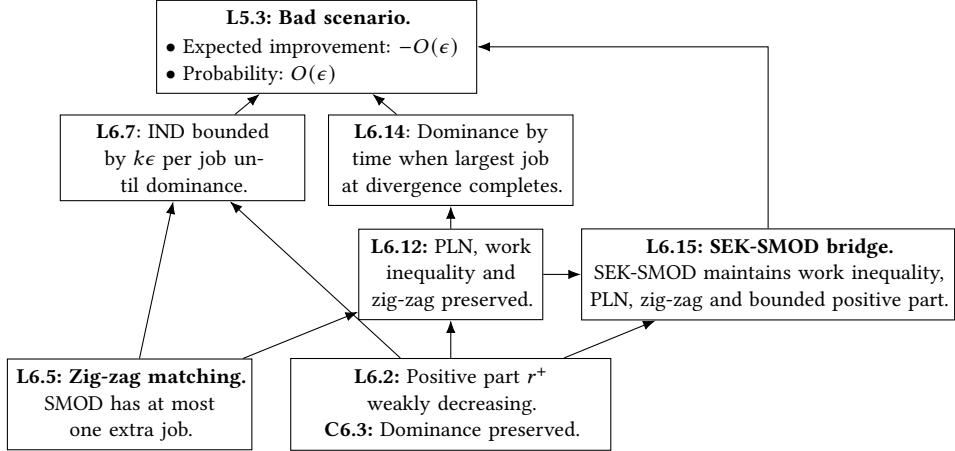


Fig. 3. Diagram of worst-case scenario arguments. The positive part (Definition 6.1), zig-zag matching (Definition 6.4), and PLN (Definition 6.11) are defined later in this section; and “work inequality” means that  $W^A(t) \leq W^{SRPT-k}$ , where policy  $A$  can be SMOD or SEK-SMOD depending on the context.

**Definition 6.1.** Define  $r^+(t)$ , the *positive part* of the difference in work between the SMOD and SRPT- $k$  systems, to be

$$r^+(t) := \sum_i (b_i^{SMOD}(t) - b_i^{SRPT-k}(t))^+.$$

Now, we can prove our first monotonicity result.

**LEMMA 6.2 (POSITIVE PART WEAKLY DECREASING).** *In the SMOD-SRPT joint system, under an arbitrary sequence of arrivals, for any pair of times  $t_1 \leq t_2$ , the positive part at time  $t_1$  upper bounds the positive part at time  $t_2$ :*

$$r^+(t_1) \geq r^+(t_2).$$

**PROOF.** There are three ways the SMOD-SRPT joint system can change: Jump change on arrival, jump change on completion, and continuous change due to work being completed. We show that each of these changes maintains the monotonicity property by carefully analyzing their effect on the remaining size and index of each of the jobs present right before and right after the event happens. The proof is intuitive once we split into the three possible events described above, so we provide the details in Appendix D.  $\square$

As a corollary, SMOD preserves the dominance relationship defined in Definition 3.1:

**COROLLARY 6.3 (DOMINANCE PRESERVED).** *If at time  $t_1$ , SMOD dominates SRPT- $k$ , then for all  $t_2 \geq t_1$ , SMOD dominates SRPT- $k$ .*

**PROOF.** The proof immediately follows from Lemma 6.2. Note that SMOD dominating SRPT- $k$  is equivalent to  $r^+ = 0$ . Therefore at time  $t_1$ ,  $r^+(t_1) = 0$  and by Lemma 6.2,  $r^+(t_2) = 0$ .  $\square$

**6.1.2 Zig-zag Matching.** Intuitively, if SMOD is only one job worse than SRPT- $k$ , it will always remain at most one job worse.

**Definition 6.4.** Define the SMOD and SRPT- $k$  systems to be *zig-zag matched* if:

$$\forall i, b_i^{SMOD} \leq b_{i+1}^{SRPT-k}.$$

In other words, the SMOD and SRPT- $k$  system are zig-zag matched if, after removing the largest job in the SMOD system, it dominates the SRPT- $k$  system. We now show that zig-zag matching is always preserved.

**LEMMA 6.5 (ZIG-ZAG).** *If at time  $t_1$ , the systems are zig-zag matched, then at all times  $t_2 \geq t_1$ , the systems will remain zig-zag matched.*

*As a result, the SMOD system has at most one additional job compared to the SRPT- $k$  system:  $N^{SMOD}(t_2) - N^{SRPT-k}(t_2) \leq 1$ .*

**PROOF.** First, we show that zig-zag matching implies that there is at most one more job in the SMOD system than in the SRPT- $k$  system. For contradiction, suppose that the SMOD system has at least two more jobs than the SRPT- $k$  system. In this case, indices 1 and 2 in the SRPT- $k$  vector are padding indices, with  $b_2^{SRPT-k} = 0$ . On the other hand, there are no padding indices in the SMOD vector, so  $b_1^{SMOD} > 0$ . This violates zig-zag matching. Thus, zig-zag matching implies that there is at most one more job in the SMOD system than in the SRPT system.

Now, we turn to proving that zig-zag matching is preserved. As in the proof of Lemma 6.2, there are three different ways that the system can change: jump change on arrivals, jump change in service, and continuous change in service. The rest of the proof carefully analyzes the job indices affected by each event, similarly to Lemma 6.2. We provide the details in Appendix E.  $\square$

**6.1.3 Integrated difference per job.** We bound SMOD's IND relative to SRPT- $k$  by examining the time when SMOD achieves dominance, and the difference after dominance.

**Definition 6.6.** Define the total future integrated number difference  $\Delta(t, \infty)_{SRPT-k, SMOD}$  between the SMOD and SRPT- $k$  system as follows:

$$\Delta_{SRPT-k, SMOD}(t, \infty) := \int_{u=t}^{\infty} N^{SRPT-k}(u) - N^{SMOD}(u) du.$$

Additionally, define  $A(t_1, t_2)$  to be the number of arrivals over the interval of time  $[t_1, t_2]$ .

With the above worst case results, we can bound the total future integrated number difference in terms of the number of arrivals:

**LEMMA 6.7.** *If at time  $t_1$  the SMOD system has a positive part  $r^+(t_1) \leq \epsilon$  and zig-zag matching, and at some later time  $t_{dom}$  the SMOD system dominates the SRPT- $k$  system, then the IND over the entire future after  $t_1$  is lower bounded by  $k\epsilon$  times the number of jobs that are present at time  $t_1$  or arrive in the interval  $[t_1, t_{dom}]$ . Formally,*

$$\Delta_{SRPT-k, SMOD}(t_1, \infty) \geq -k\epsilon(N^{SMOD}(t_1) + A(t_1, t_{dom})).$$

**PROOF.** Note that the integrand of  $\Delta_{SMOD, SRPT-k}(t, \infty)$  in Definition 6.6, namely  $N^{SRPT-k}(u) - N^{SMOD}(u)$ , simply measures the number of excess jobs in one system as compared to the other. As we want to lower bound  $\Delta_{SMOD, SRPT-k}(t_1, \infty)$ , we focus on upper bounding  $N^{SMOD}(u) - N^{SRPT-k}(u)$ . By Lemma 6.5, this difference is at most 1 at all times. We want to bound how frequently this difference equals 1.

Let us focus on the smallest job in the SMOD system at any time  $t \geq t_1$  such that  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$ . At such a time, there must be a padding index at the SRPT- $k$  system, so  $b_1^{SRPT-k} = 0$ . Thus, a positive difference must exist in the first index:  $r^+(t) \geq b_1^{SMOD}$ . But by Lemma 6.2,  $r^+(t) \leq r^+(t_1) \leq \epsilon$ .

Therefore, the smallest job in the SMOD system must have a remaining size of at most  $\epsilon$  whenever  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$ . Note that this job also receives service under SMOD at rate  $1/k$ . The first index is definitely in the eligible set, and a positive difference  $b_1^{SMOD} - b_1^{SRPT-k}$  must exist.

Among all positive-difference indices in the eligible set, the first index has the least remaining size. Thus, by Definition 3.3, the first index is guaranteed to be served.

Any given job can spend at most  $k\epsilon$  time in service with remaining size at most  $\epsilon$ .

Thus, the total amount of time  $t \geq t_1$  for which  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$  is bounded by  $k\epsilon$  times the number of distinct jobs that are served while  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$ . Note that after time  $t_{dom}$ ,  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$  never again occurs, by Corollary 6.3. Then, the only jobs that can be served while  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$  are jobs which are either present at time  $t_1$  or arrive before  $t_{dom}$ . Thus  $N^{SMOD}(t_1) + A(t_1, t_{dom})$  jobs are eligible to be served while  $N^{SMOD}(t) - N^{SRPT-k}(t) = 1$ , each for at most  $k\epsilon$  time.  $\square$

We have another minor lemma, based on the total negative difference  $r^-$ , defined similarly to the total positive difference:

**Definition 6.8.** Define  $r^-(t)$ , the *negative part* of the difference in work between the SMOD and SRPT- $k$  systems, to be

$$r^-(t) := \sum_i |\min(b_i^{SMOD}(t) - b_i^{SRPT-k}(t), 0)|.$$

This minor lemma covers the integrated difference after SMOD dominates SRPT- $k$ .

**LEMMA 6.9.** If at time  $t_1$ , SMOD dominates SRPT- $k$ , then for any times  $t_2 > t_1$ , the IND is lower bounded:

$$\int_{u=t_1}^{t_2} N^{SRPT-k}(u) - N^{SMOD}(u) du \geq k(r^-(t_1) - r^-(t_2)).$$

**PROOF.** Note that arrivals and completions do not change  $r^-$ . Note also that if the same number of jobs are served in both the SMOD and SRPT- $k$  systems,  $r^-$  also does not change. Thus whenever  $r^-$  decreases, less jobs are being served in the SMOD system than in the SRPT- $k$  system. As a result, less jobs must be present in the SMOD system than in the SRPT- $k$  system. In particular, the rate of decrease of  $r^-$  is lower bounded by the difference in number of jobs in the two systems:

$$\frac{d}{dt} r^-(t) \geq \frac{N^{SMOD}(t) - N^{SRPT-k}(t)}{k}.$$

Rearranging and integrating completes the proof.  $\square$

**6.1.4 Work inequality & Positive-less-than-negative.** We now show a form of work monotonicity: Under conditions that we specify below, the SMOD system always has at most as much work present as the SRPT- $k$  system, that is,  $W^{SMOD} \leq W^{SRPT-k}$ . The preservation of  $W^{SMOD} \leq W^{SRPT-k}$  is key to our result, and the core of the success of the SMOD policy. To prove this work monotonicity property, we first define another preserved property.

**Definition 6.10.** An index  $i$  is called a *negative-diff index* at time  $t$  if  $b_i^{SMOD}(t) < b_i^{SRPT-k}(t)$ . Similarly,  $i$  is a *positive-diff index* at time  $t$  if  $b_i^{SMOD}(t) > b_i^{SRPT-k}(t)$ .

Now we can define the PLN property:

**Definition 6.11.** The SMOD and SRPT- $k$  joint system has the *positive-less-than-negative (PLN)* property if every positive-diff index is smaller than every negative-diff index.

Note as a special case that if there are no negative-diff indices, PLN holds, and if there are no positive-diff indices, PLN holds. Intuitively, a positive-diff index  $i$  indicates that the  $i$ -th job has a larger remaining size in the SMOD system, and negative-diff implies that the corresponding job has less remaining size in the SMOD system. Hence, the PLN property says that SMOD system has

more work concentrated in the smallest indices compared to the SRPT- $k$  system. Note in particular that if SMOD dominates SRPT- $k$ , there are no positive-diff indices, and PLN holds.

Note that if the PLN property holds, then either  $N^{SMOD}(t) \geq N^{SRPT-k}$ , or SMOD dominates SRPT- $k$ . To see why, note that if a positive-diff index exists, then index 1 cannot be a negative-diff index, and specifically cannot be a padding index in the SMOD system. If there is no padding index, then  $N^{SMOD} \geq N^{SRPT-k}$ . If no positive index exists, this is the definition of dominance.

We now show that if all of PLN and  $W^{SMOD} \leq W^{SRPT-k}$  and zig-zag matching hold at some point in time, this joint property is preserved.

**LEMMA 6.12.** *If at time  $t_1$ , the systems have the PLN property and  $W^{SMOD}(t_1) \leq W^{SRPT-k}(t_1)$  and zig-zag matching, then at all times  $t_2 \geq t_1$ , the systems continue to have the same relationship: the PLN property,  $W^{SMOD}(t_2) \leq W^{SRPT-k}(t_2)$  and zig-zag matching all hold.*

**PROOF.** Note that zig-zag matching is preserved by Lemma 6.5, so we focus on establishing the other two properties, using zig-zag matching for the proof.

We know  $W^{SMOD}(t_1) \leq W^{SRPT-k}(t_1)$ , which is equivalent to  $\sum_i b_i^{SMOD}(t_1) \leq \sum_i b_i^{SRPT-k}(t_1)$ . With some algebraic manipulation, this inequality is equivalent to

$$\sum_i (b_i^{SMOD}(t_1) - b_i^{SRPT-k}(t_1))^+ \leq \sum_j (b_j^{SRPT-k}(t_1) - b_j^{SRPT-k}(t_1))^+.$$

The left-hand side summation terms are nonzero for the positive-diff indices, and the right-hand side, the negative-diff indices.

If there is no positive-diff index at time  $t_1$ , SMOD dominates SRPT- $k$  at time  $t_1$ , and by Corollary 6.3, dominance is preserved at every time  $t_2 \geq t_1$ . Note that dominance implies both the PLN property and  $W^{SMOD} \leq W^{SRPT-k}$ , so if there is no positive-diff index at time  $t_1$ , the lemma must hold.

On the other hand, if there is a positive-diff index, the PLN property tells us that the first index is not a negative-diff index. Therefore, we know that  $b_1^{SMOD}(t_1) > 0$ . If  $b_1^{SRPT-k}(t_1) = 0$ , then a padding index exists and  $N^{SMOD}(t_1) > N^{SRPT-k}(t_1)$ . If  $b_1^{SRPT-k}(t_1) > 0$ , then  $N^{SMOD}(t_1) = N^{SRPT-k}(t_1)$ , i.e. no padding. Hence,  $N^{SMOD}(t_1) \geq N^{SRPT-k}(t_1)$ .

Additionally, the zig-zag property holds at time  $t_1$ , and Lemma 6.5 implies  $N^{SMOD}(t_1) \leq N^{SRPT-k}(t_1) + 1$ . Hence, there are two possible values for  $N^{SMOD}$ :  $N^{SRPT-k}$  and  $N^{SRPT-k} + 1$ . As a result, SMOD will always serve at least as many jobs as SRPT- $k$ , so  $W^{SMOD} \leq W^{SRPT-k}$  is preserved if PLN is preserved. The proof that PLN is preserved follows by careful analysis of the effects on the indices in the SMOD and SRPT- $k$  systems of the following events: arrivals, continuous service, completions and a difference reaching zero. The approach is similar to the proof of Lemma 6.2 and we present it in Appendix F.  $\square$

**6.1.5 Time of Dominance and Maximum Size Job.** We bound the time at which SMOD achieves dominance relative to SRPT- $k$ , relative to the completion time of the initially-largest job in the SRPT- $k$  system.

**Definition 6.13.** At a given time  $t$ , we consider the *maximum index*  $i_*(t)$  for which  $b_i^{SMOD}(t) \neq b_i^{SRPT-k}(t)$ . If there is no such index, we consider  $i_*(t)$  to be undefined. We tag the job at index  $i_*(t)$  in the SRPT- $k$  system as the *max-t job* and denote it by  $m_t$ . Note that this job's index in the SRPT- $k$  queue will change throughout its time in that system. Let  $m_t(t_1)$  denote the  $m_t$ 's job's index in the SRPT- $k$  queue at time  $t_1 \geq t$ .

With this definition, we can state our final major lemma, which bounds the duration until the SMOD system dominates the SRPT- $k$  system.

**LEMMA 6.14.** Suppose that at time  $t_1$  the systems obey the PLN property, zig-zag matching, and that  $W^{SMOD}(t_1) \leq W^{SRPT-k}(t_1)$ . Then, at the time the max- $t_1$  job  $m_{t_1}$  completes in the SRPT- $k$  system, the SMOD system dominates the SRPT- $k$  system.

**PROOF.** Let  $t_{comp}$  denote the time at which the max- $t_1$  job  $m_{t_1}$  completes in the SRPT- $k$  system. We show that at all times  $t_2 \in [t_1, t_{comp}]$ , if the SMOD system does not dominate the SRPT- $k$  system, the largest index  $i_*(t_2)$  for which the two systems differ is bounded by  $m_{t_1}(t_2)$ , the index of the max- $t_1$  job at time  $t_2$ . Thus, when the max- $t_1$  job completes at time  $t_{comp}$ , either the SMOD system dominates the SRPT- $k$  system, or there is no index where the systems differ. It suffices to show that at all times  $t_2 \in [t_1, t_{comp}]$ , either SMOD dominates SRPT- $k$  or  $i_*(t_2) \leq m_{t_1}(t_2)$ .

First, note that by Lemma 6.12, at all times  $t_2 \in [t_1, t_{comp}]$ , either SMOD dominates SRPT- $k$ , or the systems continue to satisfy PLN,  $W^{SMOD}(t_2) \leq W^{SRPT-k}(t_2)$  and zig-zag matching.

If SMOD does not dominate SRPT- $k$  at time  $t_2$ , e.g. if a positive-diff index exists, then because  $W^{SMOD}(t_2) \leq W^{SRPT-k}(t_2)$ , a negative-diff index must exist. By the PLN property we know that the index  $i_*(t_2)$  must be a negative-diff index. Then, we assume that at all times prior to  $t_2$ , we have  $i_*(t_2) \leq m_{t_1}(t_2)$ , and we seek to demonstrate that the property is maintained. The rest of the proof follows immediately after considering the effects of the following events in the system: jump change on arrival, jump change on a completion, continuous service, and a diff reaching zero. We provide the details in Appendix G.  $\square$

## 6.2 SEK-SMOD-specific worst-case results

Using the SMOD monotonicity and preservation properties proven in Section 6.1, we prove SEK-SMOD properties in the following lemma.

**LEMMA 6.15.** At all times, the SEK-SMOD scheduling policy maintains the following properties relative to the coupled SRPT- $k$  system:

**Work inequality:** The SEK-SMOD system always has at most as much work present as the SRPT- $k$  system:  $W^{SEK-SMOD} \leq W^{SRPT-k}$ .

**Zig-zag:** The SEK-SMOD system always exhibits zig-zag matching, defined in Definition 6.4, with respect to the SRPT- $k$  system.

**PLN:** The SEK-SMOD system always exhibits the positive-less-than-negative property, defined in Definition 6.11, with respect to the SRPT- $k$  system.

**Positive part bound:** The SEK-SMOD system always has a bounded positive part  $r^+ \leq \epsilon$  with respect to the SRPT- $k$  system.

**PROOF.** In Lemmas 6.2 and 6.12, we showed that the SMOD policy maintains the work inequality, the zig-zag property, the PLN property, and has monotonically decreasing positive part  $r^+$ . When the two systems are merged, all four properties hold trivially. Thus, it suffices to show that all four properties hold throughout the SEK period.

To demonstrate these properties, we refer to the definition of the SEK policy in Definition 3.2. Note that the  $k-1$  smallest jobs at the divergence point received the same service in both systems, so we may focus only on the largest two jobs at the divergence point. Let  $b_1, b_k, b_{k+1}$  be the remaining sizes of the smallest, second-largest, and largest jobs at the divergence point. After  $t$  time since divergence, these two largest jobs' remaining sizes are  $[b_k - (t - b_1)^+, b_{k+1} - t]$  under SEK, and  $[b_k - t, b_{k+1} - (t - b_1)^+]$  under SRPT- $k$ . We use these remaining sizes to demonstrate the four properties. Note that either of  $b_k - (t - b_1)^+$  and  $b_{k+1} - t$  may have the largest remaining size at a given time  $t$ . We cover both possibilities.

**Work inequality:** We show that, until the time when the SEK policy finishes its final job, SEK always serves at least as many jobs as SRPT- $k$ , implying the work inequality. This is equivalent

to showing that the  $k$ th completion time under SEK, the first of the two largest jobs to complete, is no sooner under SEK than under SRPT- $k$ . In the SEK system, the second-largest job finishes at time  $t = b_1 + b_k$ , and the largest job finishes at time  $b_{k+1}$ . In the SRPT- $k$  system, the second-largest job finishes at time  $b_k$ , and the largest job finishes at time  $b_1 + b_{k+1}$ . Note that  $b_1 + b_k \geq b_k$ , and  $b_{k+1} \geq b_k$ , so these final two jobs under SEK finish after the second-to-last completion under SRPT- $k$ . This confirms the work inequality.

**Zig-zag:** We must ensure that neither of the two largest jobs' remaining sizes under SEK exceed that of the largest job under SRPT- $k$ . But note that  $b_k - (t - b_1)^+ \leq b_{k+1} - (t - b_1)^+$ , and  $b_{k+1} - t \leq b_{k+1} - (t - b_1)^+$ , confirming the zig-zag property.

**PLN:** The only indices that may have nonzero difference are those corresponding to the two largest jobs. The second-largest job has a positive difference (larger under SEK) and the largest job has a negative difference (smaller under SEK). Thus, the PLN property holds.

**$r^+$  at most  $\epsilon$ :** The only positive difference index is that of the second-largest job. This difference is at most  $(b_k - (t - b_1)^+) - (b_k - t) \leq b_1 \leq \epsilon$ .  $\square$

## 7 Numerical study

We have proven that SEK-SMOD outperforms SRPT- $k$  at all loads, with appropriate parameterization. In this section, we numerically study the Practical SEK policy, defined in Definition 7.1. We demonstrate that Practical SEK achieves significant, reliable improvements over SRPT- $k$ , across a wide range of distributions  $S$ , loads  $\rho$ , and numbers of servers  $k \geq 2$ . In Section 7.1 we compare Practical SEK with the three scheduling policies which had previously been shown to achieve optimal mean response time in heavy traffic: SRPT, PSJF (Preemptive Shortest Job First) and RS (Remaining Size times Original Size) [8]. We start with three baseline size distributions in Section 7.1, and explore a wide variety of higher-variance distributions in Section 7.2, where Practical SEK's benefit is empirically largest compared to SRPT- $k$ .

In Section 7.3 we explore settings with more servers,  $k \in \{2, 3, \dots, 6\}$ . We show that the exponentially-diminishing improvement lower bound in Lemma 5.2, with parameters defined in Definition 3.5, is a proof artifact: Practical SEK's improvement over SRPT- $k$  remains significant even as  $k$  increases, decreasing roughly linearly with  $k$ , not exponentially. We also introduce the SEK- $n$  variant of the Practical SEK policy, which is better suited to settings with more servers, defined in Definition 7.2 and explored in Section 7.3.

Finally, in Section 7.4, we show that Practical SEK's improvement over SRPT- $k$  remains significant even if the scheduler only has access to estimates of job size, not exact job size information, though estimates must be somewhat accurate for SEK to be useful.

Before presenting the results, we properly define the Practical SEK policy, a simplified single-parameter version of the SEK policy defined in Definition 3.2.

*Definition 7.1.* Practical SEK has a single parameter  $\epsilon > 0$ . Practical SEK serves the  $k$  jobs of least remaining size (matching SRPT- $k$ ), except when the following scenario arises:

- (1) There are exactly  $k + 1$  jobs present,
- (2)  $k$  of the jobs have remaining size less than  $\epsilon$ , and
- (3) the largest job has remaining size greater than  $\epsilon$ .

In this case, Practical SEK serves the  $k - 1$  jobs of least remaining size, and the final job of largest remaining size until an arrival or completion occurs.

For larger numbers of servers  $k \geq 3$ , we define the SEK- $n$  policy, which generalizes the situations in which jobs can be reordered. We numerically explore SEK- $n$  in Section 7.3.

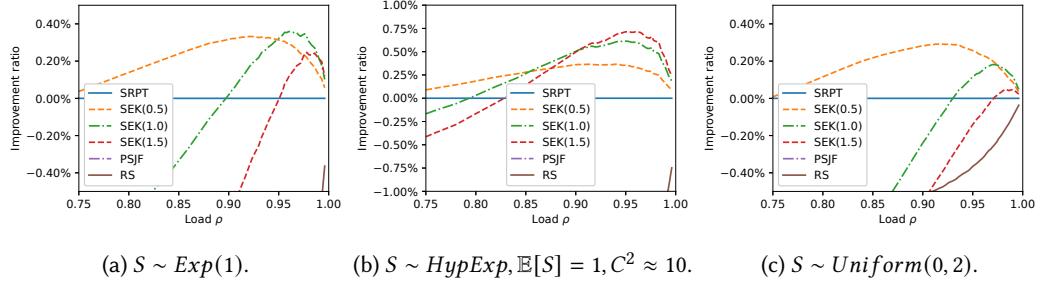


Fig. 4. Improvement of SEK over SRPT- $k$ , PSJF- $k$ , and RS- $k$ , under varying distributions  $S$  and threshold parameters  $\epsilon \in \{0.5, 1, 1.5\}$ . Improvement ratio compares each policy to SRPT- $k$ . Loads simulated:  $\rho \in [0.75, 0.996]$ ,  $k = 2$  servers.  $10^7$  arrivals per data point.

*Definition 7.2.* SEK- $n$  has two parameters,  $\epsilon > 0$  and  $n \geq 1$ . SEK-1 matches Practical SEK defined in Definition 7.1. For general  $n$ , SEK- $n$  serves the  $k$  jobs of least remaining size (matching SRPT- $k$ ), except when the following scenario arises:

- (1) There are at most  $k + n$  jobs present,
- (2)  $k$  of the jobs have remaining size less than  $\epsilon$ , and
- (3) all other jobs have remaining size greater than  $\epsilon$ .

In this case, SEK- $n$  serves the  $k - 1$  jobs of least remaining size, and the job of  $(k + 1)$ st least remaining size until an arrival or completion occurs, skipping over the job of  $k$ th least remaining size.

### 7.1 Comparison with SRPT, PSJF, and RS

In Fig. 4, we examine three values of  $\epsilon$  parameter: 0.5, 1, and 1.5, for three size distributions with mean  $\mathbb{E}[S] = 1$ : An exponential distribution in Fig. 4a, a hyperexponential distribution with squared coefficient of variation  $C^2 \approx 10$  in Fig. 4b, and a uniform distribution in Fig. 4c. In this section, we focus on the  $k = 2$  server, perfect information case. See Section 7.3 for  $k > 2$  servers, and Section 7.4 for approximate size information.

We compare these policies against the three policies which had previously been shown to achieve optimal mean response time in heavy traffic: SRPT- $k$ , PSJF- $k$ , and RS- $k$  [8].

In these figures, we plot the empirical “improvement ratio” metric  $1 - \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SRPT-}k}]}$ , where  $\pi$  represents the SEK policy for the three values of  $\epsilon$  specified above, or the SRPT- $k$ , PSJF- $k$ , and RS- $k$  policies as baselines. Note that values above 0 indicate improvement upon SRPT- $k$ , and values below 0 indicate worse performance than SRPT- $k$ .

Note that RS- $k$  never outperforms SRPT- $k$ , and is barely visible in Fig. 4a and Fig. 4b, while PSJF- $k$  is not visible in any of the plots. To make PSJF- $k$  visible, we must expand the vertical axis by a factor of 15-30×, as shown in Fig. 5.

Returning to Fig. 4 and examining the different threshold parameters  $\epsilon$  for the Practical SEK policy, we see that an epsilon parameter of 0.5 achieves some improvement over SRPT- $k$  across the entire range of loads simulated, with a low sensitivity to job size distribution variability, while the larger thresholds  $\epsilon = 1$  and  $\epsilon = 1.5$  achieve higher peak improvement ratios, with a peak improvement ratio of 0.32% for  $\epsilon = 1$  in the exponential case in Fig. 4a, 0.67% for  $\epsilon = 1.5$  in the hyperexponential case in Fig. 4b, and 0.20% in the uniform case in Fig. 4c.

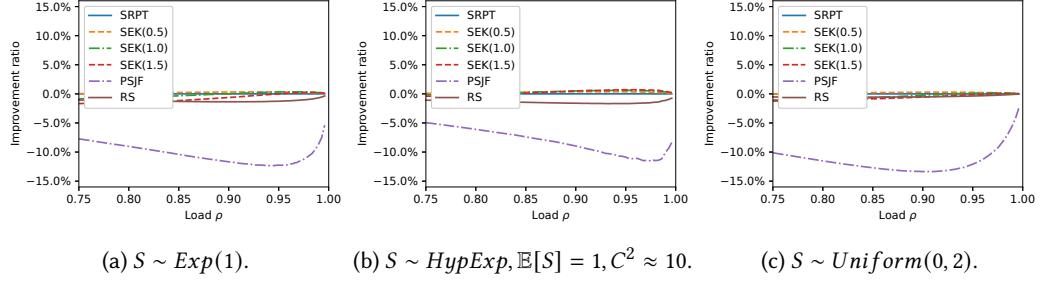


Fig. 5. Improvement of SEK over SRPT- $k$ , PSJF- $k$ , and RS- $k$ , under varying distributions  $S$  and threshold parameters  $\epsilon \in \{0.5, 1, 1.5\}$ , with an expanded vertical axis. Improvement ratio compares each policy to SRPT- $k$ . Loads simulated:  $\rho \in [0.75, 0.996]$ ,  $k = 2$  servers.  $10^7$  arrivals per data point.

$C^2$	$\rho_{high}$	Best Improvement	Best $\rho$	Best $\epsilon$	$C^2$	$\rho_{high}$	Best Improvement	Best $\rho$	Best $\epsilon$
2	0.1	0.4828%	0.97	1	10	0.7	0.7756%	0.94	3
2	0.3	0.4982%	0.97	1	10	0.9	0.5106%	0.93	3
2	0.5	0.4912%	0.96	1	20	0.1	0.7524%	0.98	1.5
2	0.7	0.4968%	0.95	1	20	0.3	<b>0.9040%</b>	0.98	3
2	0.9	0.4564%	0.95	1	20	0.5	0.8985%	0.96	3
4	0.1	0.6145%	0.98	1.5	20	0.7	0.6999%	0.94	3
4	0.3	0.6381%	0.98	1.5	20	0.9	0.3707%	0.9	3
4	0.5	0.6282%	0.97	1.5	40	0.1	0.7468%	0.99	2
4	0.7	0.6323%	0.96	2	40	0.3	0.7670%	0.96	1.5
4	0.9	0.5084%	0.93	1.5	40	0.5	0.8146%	0.95	3
10	0.1	0.8056%	0.99	2	40	0.7	0.5936%	0.93	3
10	0.3	0.7785%	0.98	3	40	0.9	0.2262%	0.9	3
10	0.5	0.8238%	0.96	3					

Table 2. Experiments varying 2-branch hyperexponential distributions  $S$ , maximized over load  $\rho$  and threshold parameter  $\epsilon$ .  $k = 2$  servers.  $10^7$  arrivals per data point.

These results for these distributions are indicative of a wider trend: Practical SEK is more beneficial, relative to SRPT- $k$ , when the job size distribution has higher variance. We explore this trend further in Section 7.2.

## 7.2 Systematic exploration of higher-variance distributions

In this section, we demonstrate that SEK demonstrates larger improvement over SRPT- $k$ , under higher variance job size distributions. We specifically focus on 2-branch hyperexponential job size distributions, with  $k = 2$  servers. For  $k > 2$  servers, see Section 7.3.

We parameterize our size distributions by squared coefficient of variability,  $C^2$ , and by the fraction of load  $\rho_{high}$  comprised by the high-mean branch of the hyperexponential distribution. We normalize the mean  $E[S] = 1$ . In Table 2, we explore job size distributions defined by all pairs of  $C^2 \in \{2, 4, 10, 20, 40\}$  and high-mean load fraction  $\in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ , across loads  $\rho \in [0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99]$ , and across SEK thresholds  $\epsilon \in [1, 1.5, 2, 3]$ . For each distribution, we give the  $(\rho, \epsilon)$  pair which maximizes the improvement ratio  $1 - \frac{\mathbb{E}[T^{\pi}]}{\mathbb{E}[T^{SRPT-k}]}$ , with the globally highest improvement ratio in bold. The highest improvement ratio we found was

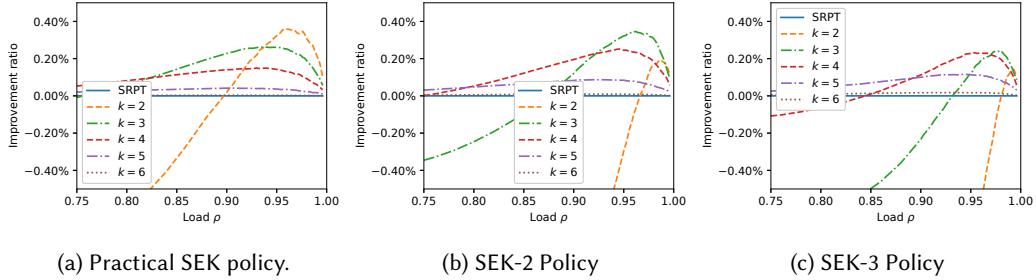


Fig. 6. Improvement of SEK over SRPT- $k$ , and the SEK2 and SEK3 variants, for  $k = 2, 3, \dots, 6$  servers. Threshold  $\epsilon = 1$ . Loads simulated:  $\rho \in [0.75, 0.996]$ . Job size:  $S \sim \text{Exp}(1)$ .  $10^7$  arrivals per data point.

a 0.90% improvement, with  $C^2 = 20$  and  $\rho_{high} = 0.3$ , with  $\rho = 0.98$  and  $\epsilon = 3$ . Further exploration would likely uncover even larger improvement ratios.

Consistent trends are visible in Table 2: Larger improvements are associated with higher  $C^2$  and lower  $\rho_{high}$ , and the best threshold  $\epsilon$  increases with larger  $C^2$  and larger  $\rho_{high}$ .

### 7.3 Larger number of servers $k$ , and policy variant SEK- $n$

In this section, we examine the improvement of the SEK policy over the SRPT- $k$  policy for  $k = 2, 3, \dots, 6$  servers. In Fig. 6a, we compare the standard Practical SEK policy with SRPT- $k$  over this range of servers. Here we see that for  $k \geq 3$  servers, improvement over SRPT- $k$  is more consistent across a range of loads, extending to all loads simulated. However, peak improvement is smaller as  $k$  increases. Note however that the exponential decrease in  $k$  of the improvement lower bound proven in Lemma 5.2, with parameters given in Definition 3.5, is overly pessimistic: Even at  $k = 5$  servers, improvement is still significant.

To enlarge peak improvement for more servers  $k$ , we introduce a variant of the Practical SEK policy, which we call the SEK- $n$  family of policies, which we defined in Definition 7.2. Under Practical SEK, we serve the  $k + 1$ st largest job in front of the  $k$ th largest job if and only if there are  $k + 1$  jobs in the system, of which  $k$  have remaining size below  $\epsilon$  and 1 has remaining size above  $\epsilon$ . Under SEK- $n$ , we generalize the cases in which we serve the  $k + 1$ st largest job in front of the  $k$ th largest job to allow up to  $n$  jobs with remaining size above  $\epsilon$ , while still requiring exactly  $k$  jobs with remaining size below  $\epsilon$  to be present.

The intuition behind the SEK- $n$  policy is similar to that of the baseline SEK policy: We find situations where the SRPT- $k$  policy would waste capacity by prematurely emptying one or more servers by completing the smallest jobs, and instead serve a larger job to more efficiently utilize capacity. Note that this intuition only applies when  $n < k$ , and we see in Fig. 6 that SEK- $n$  is most beneficial when  $n < k$ .

In particular, in Fig. 6b and Fig. 6c, we show the improvement of the SEK-2 and SEK-3 policies over the the SRPT- $k$  policy, again for  $k = 2, 3, \dots, 6$  servers. For  $k = 3$ , the peak improvement is largest under SEK-2, with similar peak improvement ratio as for SEK-1 with  $k = 2$  servers.  $k = 4$  similarly shows largest improvement under SEK-2, while for  $k = 5$  and  $k = 6$ , the largest improvement is found for SEK-3.

### 7.4 Estimated job sizes

Previously, we have focused on the case where the scheduler has access to exact size information. In this section, we consider the case where the scheduler only has access to estimated size information.

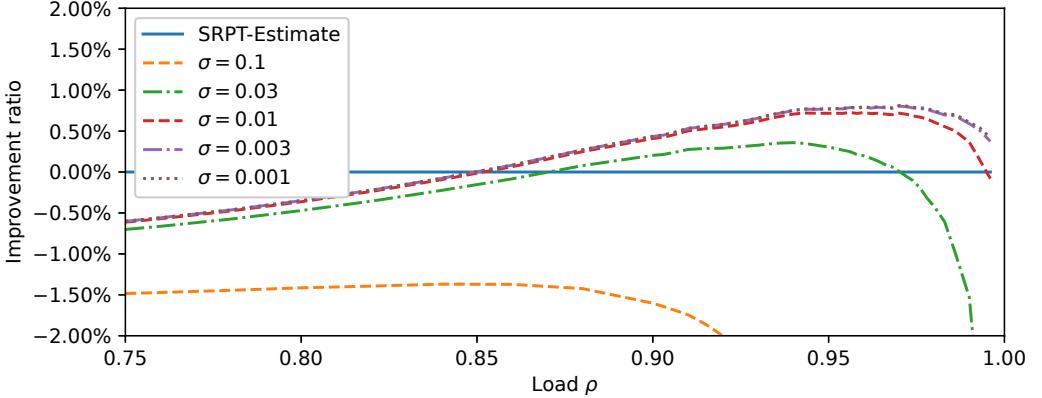


Fig. 7. Improvement of SEK-Estimate over SRPT-Estimate, for normally distributed error  $D$  with standard deviation  $\sigma$ . Job size distribution  $S$  is hyperexponential with  $C^2 \approx 10$  and  $E[S] = 1$ . Threshold  $\epsilon = 2$ . Loads simulated:  $\rho \in [0.75, 0.996]$ ,  $k = 2$  servers.  $10^7$  arrivals per data point.

In this setting, when each job arrives, the scheduler has access to a size estimate  $E := (S + D)^+$ , where  $D$  is an i.i.d. error margin. As our baseline policy, we consider the SRPT-Estimate policy, which prioritizes jobs according to the remaining size estimate  $E - a$ , where  $a$  is the job's age (i.e. time in service). Note that more refined or optimized remaining size estimates are known in the literature [24], but this estimate makes a useful comparison point for testing SEK. Specifically, we compare SRPT-Estimate against the SEK-Estimate policy, which plugs the same estimate into Practical SEK policy defined in Definition 7.1.

In Fig. 7, we empirically demonstrate that SEK-Estimate can improve upon SRPT-Estimate, for accurate enough estimates. Specifically, we consider the estimate distribution  $D \sim \text{Normal}(0, \sigma)$ , with the standard deviation  $\sigma \in \{0.1, 0.03, 0.01, 0.003, 0.001\}$ . We use a choicer threshold  $\epsilon = 2$  than in Section 7.1, which we find produces better empirical results with estimated job sizes.

If errors are sufficiently large, e.g.  $\sigma = 0.1$ , SEK-Estimate is worse than SRPT-Estimate. However, as errors shrink, SEK-Estimate produces increasingly large improvements over SRPT-Estimate, converging to the improvement margin in the perfect-information case.

Intuitively, the benefits of SEK over SRPT simply require SEK to be able to identify states in which near-future work efficiency is more important than near-future completion speed. While sufficiently large estimation errors interfere with this identification enough to make SEK-Estimate ineffective, moderate to high accuracies of 3% relative error or less are sufficient for SEK-Estimate to identify beneficial interchange states.

## 8 Conclusion

We demonstrate the first scheduling policy which provably improves on the mean response time of SRPT- $k$  in the M/G/ $k$  queue. We prove in Theorem 4.1 that our SRPT-Except- $k+1$  & Modified SRPT (SEK-SMOD) policy has lower mean response time than SRPT- $k$  at all loads and under all job size distributions, despite SRPT- $k$ 's optimality results in the heavy traffic limit. These theoretical results are empirically validated in Section 7.

To prove our results, we combine hybrid worst-case-and-stochastic analysis with relative analysis. We characterize the stochastic properties of the SEK portion of the policy in Section 5 and the worst-case properties of the SMOD portion of the policy Section 6, both relative to the behavior of

the baseline SRPT- $k$  policy. We thereby demonstrate an explicit parameter choice for the SEK-SMOD policy which always achieves lower mean response time than SRPT- $k$  policy.

In Section 7, we numerically explore the improvement of a practical variant of SEK over SRPT- $k$  under a variety of job size distributions  $S$ , numbers of servers  $k$ , and under estimated job sizes, rather than perfect information. We find the SEK's improvement over SRPT- $k$  is modest, but significant, peaking at around 0.9% improvement, which recent work indicates is a substantial fraction of the possible improvement from SRPT- $k$  to the optimal  $k$ -server policy [26]. In Section 7.3, we also introduce a policy variant, SEK- $n$ , which expands the situations in which a larger job is run ahead of a smaller job to improve performance for moderately larger numbers of servers  $k \geq 3$ .

An important direction for future work lies in optimization-based and learning-based search for additional improvement in mean response time over SRPT- $k$ . The continuous-time, continuous-state, infinite-dimensional nature of the known-size M/G/k setting make it hard to learn or optimize effective policies, but the lower-dimensional nature of SEK-type policies may be easier to search and optimize over. Future work could also seek to improve relative to the Gittins- $k$  policy [22], allowing more complex partial information settings.

## References

- [1] Shany Azaria, Boaz Ronen, and Noam Shamir. 2023. Justice in time: A theory of constraints approach. *Journal of Operations Management* 69, 7 (2023), 1202–1208. doi:10.1002/joom.1234
- [2] Shany Azaria, Boaz Ronen, and Noam Shamir. 2024. Alleviating court congestion: The case of the Jerusalem district court. *INFORMS Journal on Applied Analytics* 54, 3 (2024), 267–281.
- [3] Robert L Bray, Decio Coviello, Andrea Ichino, and Nicola Persico. 2016. Multitasking, multiarmed bandits, and the Italian judiciary. *Manufacturing & Service Operations Management* 18, 4 (2016), 545–558.
- [4] Richard Walter Conway, William L. Maxwell, and L. W. Miller. 1967. *Theory of scheduling*. Addison-Wesley Pub. Co., Reading, Mass.
- [5] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In *Annals of Discrete Mathematics*. Vol. 5. North-Holland Publishing Company, Amsterdam, The Netherlands, 287–326.
- [6] Isaac Grosof, Mor Harchol-Balter, and Alan Scheller-Wolf. 2022. WCFS: A new framework for analyzing multiserver systems. *Queueing Systems* 102, 1 (2022), 143–174.
- [7] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2019. Load balancing guardrails: Keeping your heavy traffic on the road to low response times. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 2 (2019), 1–31.
- [8] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2019. SRPT for multiserver systems. *ACM SIGMETRICS Performance Evaluation Review* 46, 2 (2019), 9–11.
- [9] Izzy Grosof and Ziyuan Wang. 2024. Bounds on M/G/k Scheduling Under Moderate Load Improving on SRPT- $k$  and Tightening Lower Bounds. *SIGMETRICS Perform. Eval. Rev.* 52, 2 (Sept. 2024), 24–26. doi:10.1145/3695411.3695421
- [10] Isaac Grosof, Kunhe Yang, Ziv Scully, and Mor Harchol-Balter. 2021. Nudge: Stochastically improving upon FCFS. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 2 (2021), 1–29.
- [11] Yige Hong and Ziv Scully. 2024. Performance of the Gittins policy in the G/G/1 and G/G/k, with and without setup times. *Performance Evaluation* 163 (2024), 102377. doi:10.1016/j.peva.2023.102377
- [12] Prasad A Kulkarni and Jay Fuller. 2011. JIT Compilation Policy on Single-Core and Multi-core Machines. In *2011 15th Workshop on Interaction between Compilers and Computer Architectures*. 54–62. doi:10.1109/INTERACT.2011.9
- [13] Julian Kölleström. 1974. Heavy traffic theory for queues with several servers. I. *Journal of Applied Probability* 11, 3 (1974), 544–552. doi:10.2307/3212698
- [14] Julian Kölleström. 1979. Heavy traffic theory for queues with several servers. II. *Journal of Applied Probability* 16, 2 (1979), 393–401. doi:10.2307/3212906
- [15] Stefano Leonardi and Danny Raz. 2007. Approximating total flow time on parallel machines. *Journal of computer and system sciences* 73, 6 (2007), 875–891.
- [16] Yuan Li and David A. Goldberg. 2025. Simple and Explicit Bounds for Multiserver Queues with  $1/1-\rho$  Scaling. *Mathematics of Operations Research* 50, 2 (2025), 813–837. doi:10.1287/moor.2022.0131
- [17] John D. C. Little and Stephen C. Graves. 2008. *Little's Law*. Springer US, Boston, MA, 81–100. doi:10.1007/978-0-387-73699-0\_5

- [18] Richard Loulou. 1973. Multi-channel queues in heavy traffic. *Journal of Applied Probability* 10, 4 (1973), 769–777. doi:10.2307/3212380
- [19] Robert McNaughton. 1959. Scheduling with deadlines and loss functions. *Management science* 6, 1 (1959), 1–12.
- [20] Linus Schrage. 1968. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research* 16, 3 (1968), 687–690.
- [21] Linus E. Schrage and Louis W. Miller. 1966. The Queue M/G/1 with the Shortest Remaining Processing Time Discipline. *Operations Research* 14, 4 (1966), 670–684.
- [22] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2020. The Gittins policy is nearly optimal in the M/G/k under extremely general conditions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 3 (2020), 1–29.
- [23] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2021. Optimal multiserver scheduling with unknown job sizes in heavy traffic. *Performance Evaluation* 145 (2021), 102150.
- [24] Z Scully, I Grosof, and M Mitzenmacher. 2022. Uniform Bounds for Scheduling with Job Size Estimates. In *13th Innovations in Theoretical Computer Science Conference, ITCS*.
- [25] Andrei Sleptchenko, Aart van Harten, and Matthieu van der Heijden. 2005. An Exact Solution for the State Probabilities of the Multi-Class, Multi-Server Queue with Preemptive Priorities. *Queueing Systems* 50, 1 (01 May 2005), 81–107. doi:10.1007/s11134-005-0359-y
- [26] Ziyuan Wang and Isaac Grosof. 2025. Novel Lower Bounds on M/G/k Scheduling. *arXiv preprint arXiv:2502.08804* (2025).
- [27] Sergey Zhuravlev, Juan Carlos Saez, Sergey Blagodurov, Alexandra Fedorova, and Manuel Prieto. 2012. Survey of scheduling techniques for addressing shared resources in multicore processors. *ACM Comput. Surv.* 45, 1, Article 4 (Dec. 2012), 28 pages. doi:10.1145/2379776.2379780

Table 3. Notation for random variables and system parameters.

Notation	Meaning
$\lambda$	Mean arrival rate.
$S$	Generally distributed random variable that represents the size of an arriving job.
$k$	Number of servers. Each of them processes jobs at a fixed rate of $1/k$ .
$\rho$	Mean load of the system, that is, $\rho = \lambda \mathbb{E}[S]$ .
$\epsilon' < \epsilon$	Parameters of SEK policy. One of the conditions to trigger SEK is that $k$ of the $k+1$ jobs present have remaining sizes between $\epsilon'$ and $\epsilon$ .
$x < y$	Parameters of the SEK policy. One of the conditions to trigger SEK is that the largest job has a remaining size between $x$ and $y$ . Note that $\epsilon \leq x$ .

## A Summary of Notation

In this section, we review the notation we used to define our model and scheduling policies, and that are repeatedly used in the manuscript. The parameters and random variables associated with the M/G/ $k$  system are presented in Table 3. These are mostly generic parameters, and are organized in order of appearance.

In Table 4, we review the variables introduced to define the SEK-SMOD policy, including state descriptors and performance measures. These are organized alphabetically, with Greek letters listed at the end.

## B Details of proof of Lemma 5.3

To finalize the proof, we need to explicitly compute a lower bound for  $\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{BAD}_t]$ .

**PROOF.** Let  $t_{dom}$  be the first time after  $t$  when the SEK-SMOD system dominates the SRPT- $k$  system, or infinity if dominance never occurs. Lemma 6.7 (combined with Lemma 6.15) shows that  $\Delta_t$  is lower bounded by at most  $-k\epsilon$  times the number of jobs which arrive prior to  $t_{dom}$ , plus the number of jobs present at the time SMOD begins being used, which is at most  $k+2$ . Because  $t_{dom}$  is a stopping time with respect to the arrival process, by Wald's equation we know that the expected number of jobs which arrive prior to  $t_{dom}$  is  $\lambda \mathbb{E}[t_{dom}]$ . Thus, we know that

$$\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{BAD}_t] \geq -k\epsilon(\lambda \mathbb{E}[t_{dom} \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{BAD}_t] + k + 2). \quad (5)$$

Now, we need to bound  $t_{dom}$  for an arbitrary initial state. For this, we invoke Lemma 6.14, which (in combination with Lemma 6.15) states that by the time largest job in the SEK divergence scenario completes in the SRPT- $k$  system, both systems are merged or SEK-SMOD dominates SRPT- $k$ . Let  $j$  denote this particular job. Thus,  $t_{dom}$  is bounded by the remaining response time of this job  $j$ :

$$\mathbb{E}[t_{dom} \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{BAD}_t] \leq \mathbb{E}[T_j \mid \mathbf{bb}(t) = \mathbf{bb} \& \text{BAD}_t]. \quad (6)$$

We now apply the multiserver tagged job analysis of [8]. Let the remaining size of the largest job  $j$  at the time of divergence  $t$  be  $r$ , and let  $w$  be the relevant work at time  $t$ , that is, the total work of remaining size less than or equal to  $r$  at time  $t$ . Note that  $r \leq y$ , by the SEK divergence policy, and that  $w \leq k\epsilon + y$ , as the total relevant work consists only of the  $k$  other jobs present at divergence, plus the newly arrived job, which can contribute at most  $r \leq y$  relevant work. By the multiserver

Table 4. State descriptor notation.

Notation	Meaning
$\mathbf{b}^A(t)$	Vector of ordered remaining sizes at time $t$ in an M/G/k queue under scheduling policy $A$ , potentially augmented by $d(t)$ padding indices with respect to a policy $B$ . If $N^A(t) \geq N^B(t)$ , then $\mathbf{b}^A(t) = \mathbf{r}^A(t)$ and otherwise, $b_i^A(t) = 0$ for $i \leq d(t)$ and $b_i^A(t) = r_{i-d(t)}^A(t)$ for $i \geq d(t) + 1$ .
$\mathbf{b}^B(t)$	Definition symmetrical to $\mathbf{b}^A(t)$ .
$\mathbf{bb}(t)$	State descriptor of two M/G/k queues with coupled arrivals, one operating under policy $A$ and the other one under policy $B$ . Specifically, $\mathbf{bb}(t) = (\mathbf{b}^A(t), \mathbf{b}^B(t))$ .
$d(t)$	Number of padding indices, that is, the difference in number of jobs at time $t$ , between an M/G/k queue under policy $A$ and one under policy $B$ . In other words, $d(t) =  N^A(t) - N^B(t) $ .
IND	Integrated number difference, that is, integral of the difference in number of jobs between the SRPT- $k$ and the SEK-SMOD system.
$N^\pi(t)$	Number of jobs at time $t$ in an M/G/k queue under scheduling policy $\pi$ . Note that $N^\pi(t) =  \mathbf{r}^\pi(t) $ .
$\mathbf{r}^\pi(t)$	Vector of ordered remaining sizes at time $t$ in an M/G/k queue under scheduling policy $\pi$ . Each element $r_i(t)$ represents the remaining size of the $i$ -th smallest job in the queue (including jobs in service).
$r^+(t)$	Positive part of the difference in work between the SMOD and SRPT- $k$ systems.
$r^-(t)$	Negative part of the difference in work between the SMOD and SRPT- $k$ systems.
$T^\pi$	Response time under the scheduling policy $\pi$ .
$W^\pi(t)$	Total remaining work at time $t$ in an M/G/k queue under scheduling policy $\pi$ . Note that $W^\pi(t) = \sum_i r_i(t)$ .
$\Delta_t$	Mathematical definition of IND, that is, integral of $N^{SRPT-k}(u) - N^{SEK-SMOD}(u)$ for $u$ between times $t$ and $\tau$ .
$\Delta_{SRPT-k,SMOD}(t, \infty)$	IND from time $t$ to the future.
$\tau$	Starting at time $t$ , next time when SRPT- $k$ and SEK-SMOD have identical states.

tagged job analysis [8, eq. 5.1 and Lemma 1], we know that

$$\begin{aligned} [T_j \mid \mathbf{bb}(t) = \mathbf{bb} \text{ & } \text{BAD}_t] &\leq_{st} B_{\leq y}(kr + w), \\ \mathbb{E}[T_j \mid \mathbf{bb}(t) = \mathbf{bb} \text{ & } \text{BAD}_t] &\leq \frac{kr + w}{1 - \rho_{\leq r}} \leq \frac{(k+1)y + k\epsilon}{1 - \rho_{\leq y}}, \end{aligned}$$

where  $B_{\leq y}(\cdot)$  denotes a relevant busy period with relevancy cutoff  $y$ , and  $\rho_{\leq r} = \lambda \mathbb{E}[S \mathbb{1}\{S \leq r\}]$  is the relevant load at threshold  $r$ .

Note from Definition 3.5 that  $\epsilon < x/6$ , so we can write:

$$\mathbb{E}[T_j \mid \mathbf{bb}(t) = \mathbf{bb} \text{ & } \text{BAD}_t] \leq \frac{(k+1)y + kx/6}{1 - \rho_{\leq y}}. \quad (7)$$

Combining together (5), (6), and (7), we find that

$$\mathbb{E}[\Delta_t \mid \mathbf{bb}(t) = \mathbf{bb} \text{ & } \text{BAD}_t] \geq -k\epsilon \left( \lambda \frac{(k+1)y + kx/6}{1 - \rho_{\leq y}} + k + 2 \right).$$

Because  $c_2$  defined in Definition 3.5 is the constant multiplying  $-\epsilon$  above, the proof is complete.  $\square$

## C Details of the proof of Lemma 5.4

### C.1 Probability of a good scenario

We need to show that the probability of a good scenario is lower bounded by  $c_3$ , defined in Definition 3.5.

**PROOF.** Let  $b_{k+1}$  be the size of the largest job in divergence starting state  $\mathbf{bb}$ . The good scenario specifies that no jobs arrive in the first  $k(b_{k+1} - 2x/3)$  time, then exactly  $k$  jobs with sizes in  $[x, 2x]$  arrive during the interval  $[k(b_{k+1} - 2x/3), k(b_{k+1} - x/3)]$ , and then no jobs arrive until time  $k(b_{k+1} + 2x)$ .

The good scenario consists of three independent events, whose probabilities can be exactly quantified, using the fact that the arrival process is a Poisson process with rate  $\lambda$ :

- No arrivals in the intervals  $[0, k(b_{k+1} - 2x/3)]$  or  $[k(b_{k+1} - x/3), k(b_{k+1} + 2x)]$ , a total duration of  $k(b_{k+1} + 5x/3)$ , which has probability  $e^{-\lambda k(b_{k+1} + 5x/3)} \geq e^{-\lambda k(y + 5x/3)}$ , because  $b_{k+1} \leq y$ .
- Exactly  $k$  arrivals during the interval  $[k(b_{k+1} - 2x/3), k(b_{k+1} - x/3)]$ , a duration of  $kx/3$ , which has probability  $\frac{(\lambda kx/3)^k e^{-\lambda kx/3}}{k!}$ .
- Each of those  $k$  arrivals has size in the range  $[x, 2x]$ , which has probability  $(\mathbb{P}(S \in [x, 2x]))^k$ . Note that this probability is guaranteed to be positive by the selection of  $s$  in Lemma 5.2.

The probability that the good event occurs is the product of the probabilities of these independent events, which is lower bounded by the product of the probabilities and lower bounds provided in each bullet point. As desired, such product equals  $c_3$  defined in Definition 3.5.  $\square$

### C.2 Expected difference in a good scenario

To finalize the proof, we prove each of the claims in detail.

**PROOF.** In the rest of the proof, we analyze the effects of each of the following events in the expected  $\Delta_t$ .

**The  $k-1$  smallest jobs at time of divergence:** Because no jobs arrive in the first  $k(b_{k+1} - 2x/3)$  time after divergence, and because  $b_{k+1} \geq x > 6\epsilon$ , no jobs arrive before the  $k$  smallest jobs at time of divergence (in  $\mathbf{bb}$ ) have completed in both systems. The only difference between SEK and SRPT- $k$  at the time of divergence is whether the second-largest job is in service (under SRPT- $k$ ) or the largest job is in service (under SEK). Thus, the completion times of all of the other  $k-1$  smaller jobs are the same in both systems, resulting in no effect on  $\Delta_t$ .

**The second-largest job at time of divergence:** This job has remaining size  $b_k$  at the time of divergence, so it completes at time  $kb_k$  under SRPT- $k$  and at  $k(b_1 + b_k)$  under SEK-SMOD. Then, there is a  $kb_1$  contribution to  $\Delta_t$ .

**The largest job at time of divergence:** Because additional jobs do not arrive until time  $k(b_{k+1} - 2x/3)$  at the earliest, the largest job at time of divergence has reached a remaining size of no more than  $2x/3 + b_1$  under SRPT- $k$ , as it began service at time  $kb_1$ . Under SEK-SMOD, its

remaining size is no more than  $2x/3$ , as the job began service at the divergence point. Because  $s > 6\epsilon$ , this job is smaller than all of the arriving jobs, and thus is prioritized ahead of the arriving jobs, remaining in service. It thus finishes at the same time it would have in the absence of those arrivals, finishing  $kb_1$  time earlier in the SEK-SMOD system than in the SRPT system. This gives a  $-kb_1$  contribution to  $\Delta_t$ .

**The arriving jobs:** Because of the exclusion of further arrivals under the good scenario,  $k - 1$  of the arriving jobs enter service immediately and complete without any wait in both systems. The only exception is the job with the largest remaining size at its arrival time, which is delayed behind the job with the largest remaining size at divergence, the remaining-size- $b_{k+1}$  job. This job is the first to complete and free up a server for the largest of the arriving jobs. Thus, this largest job's response time differs in the two systems by  $k$  times the remaining size of this originally- $b_{k+1}$  size job at the time of this arrival. The originally- $b_{k+1}$  size job will be in the system at this time, as it completes at time  $kb_{k+1}$  at the earliest, after all the arrivals have occurred.

Throughout this period, the originally- $b_{k+1}$  size job has larger remaining size in the SRPT- $k$  system, by a margin of  $b_1$  remaining size. Thus, the largest arriving job finishes  $kb_1$  later in the SRPT- $k$  system than in the SEK-SMOD system.

This confirms all of the claims. Because  $b_1 \geq \epsilon' = \epsilon/2$ , we obtain the desired bound with  $c_4 = k/2$ , as defined in Definition 3.5.  $\square$

## D Details of proof of Lemma 6.2

To complete the proof, we analyze how arrivals, completions, continuous service and a difference reaching zero influence each job's index and, consequently, the positive difference  $r^+$ .

PROOF. We analyze the effects on the positive difference of each of the following events:

**Jump change on arrival:** We first show that an arrival, occurring in both queues at the same time with the same size, does not change the value of  $r^+(t)$ . Specifically, we define the times  $t^{(-)}$  and  $t^{(+)}$  to denote the left and right limits approaching a given time  $t$ :

$$r^+(t^{(-)}) := \lim_{s \rightarrow t | s \leq t} r^+(s), \quad r^+(t^{(+)}) := \lim_{s \rightarrow t | s \geq t} r^+(s).$$

Our goal is to prove that  $r^+(t^{(-)}) = r^+(t^{(+)})$ , for any arrival time  $t$ .

For concision, we write the padded vectors  $b^{SMOD}(t^{(-)})$  and  $b^{SRPT-k}(t^{(-)})$ , just before the arrival at time  $t$ , as  $p$  and  $q$  respectively. Specifically, let the remaining sizes of the SMOD system just prior to the arrival be  $p_1 \leq p_2 \leq p_3 \leq \dots$  and let the remaining sizes in the SRPT- $k$  system just prior to the arrival be  $q_1 \leq q_2 \leq q_3 \leq \dots$ . Note that some of these values may be zero, due to the padding.

Let the arriving job have size  $x$ , and let its index upon arrival in the SMOD system be  $i$  and in the SRPT- $k$  system be  $j$ . In other words,  $p_{i-1} \leq x \leq p_i$ , and  $q_{j-1} \leq x \leq q_j$ .

Note that all indices  $\ell$  less than  $\min(i - 1, j - 1)$  or more than  $\max(i, j)$  preserve the difference  $p_\ell - q_\ell$  before and after the arrival – for small indices, the arrival is irrelevant, and for large indices, each job's index is increased by 1, leaving the difference unchanged. Thus, we only need to look at the effect on  $r^+$  from indices between  $i$  and  $j$  (inclusive).

Let's consider three cases:  $i = j$ ,  $i < j$ ,  $i > j$ .

- If  $i = j$ , the arriving job has the same index in both systems, contributing 0 to  $r^+$ .
- If  $i < j$ , the arriving job of size  $x$  affects the indices between  $i$  and  $j$  as follows:

$$\begin{aligned} b^{SMOD}(t^{(-)}) : & \quad p_i, \quad p_{i+1}, \quad \dots, \quad p_{j-1}; & b^{SMOD}(t^{(+)}) : & \quad x, \quad p_i, \quad \dots, \quad p_{j-2}, \quad p_{j-1}. \\ b^{SRPT-k}(t^{(-)}) : & \quad q_i, \quad q_{i+1}, \quad \dots, \quad q_{j-1}; & b^{SRPT-k}(t^{(+)}) : & \quad q_i, \quad q_{i+1}, \quad \dots, \quad q_{j-1}, \quad x \end{aligned}$$

By definition, we have

$$r^+(t^{(-)}) = \sum_{\ell=i}^{j-1} (p_\ell - q_\ell)^+, \quad \text{and} \quad r^+(t^{(+)}) = (x - q_i)^+ + (p_{j-1} - x)^+ + \sum_{\ell=i+1}^{j-1} (p_\ell - q_{\ell+1})^+.$$

Note that, for all indices  $i' > i$  and  $j' < j$  we have  $q_{j'} \leq x \leq p_{i'}$  because our state descriptor orders the remaining sizes from smallest to largest. Thus, for each index  $\ell \in (i, j)$ , the difference before arrival  $(p_\ell - q_\ell)^+ = p_\ell - q_\ell$  is nonnegative. Likewise, for each index the difference after arrival  $(p_\ell - q_{\ell+1})^+ = p_\ell - q_{\ell+1}$  is also nonnegative. Finally, the difference  $(x - q_i)^+ = x - q_i$  and  $(p_{j-1} - x)^+ = p_{j-1} - x$ . Thus, we find that  $r^+(t^{(-)}) = r^+(t^{(+)})$ .

- Finally, if  $i > j$ , we have the reversed picture. Following a similar argument, we show that  $r^+(t^{(-)}) = r^+(t^{(+)})$  as well.

**Jump change on completions:** Similarly to the arrivals, we split into three cases: A completion where both systems have an equal number of jobs, a completion in the system with fewer jobs, and in the system with more jobs.

- In the first case, supposing that the completion happens at time  $t$ , we have the following diagram, where  $\delta$  represents the job very close to completion:

$$\begin{array}{lll} b^{SMOD}(t^{(-)}) : & \delta, & p_2, p_3, \dots \\ b^{SRPT-k}(t^{(-)}) : & q_1, & q_2, q_3, \dots \end{array} ; \quad \begin{array}{lll} b^{SMOD}(t^{(+)}) : & 0, & p_2, p_3, \dots \\ b^{SRPT-k}(t^{(+)}) : & q_1, & q_2, q_3, \dots \end{array} .$$

Note that  $q_1 > 0$ , because padding only occurs to a system with less jobs. The 0 appears in the SMOD system due to that padding. In this step, we ignore the change in the remaining sizes of other jobs, as we handle this case separately in the proof. Here we focus on the effect of the completion.

The state space changes continuously in each index, so  $r^+$  doesn't change as a result of this kind of completion. We showed the effect of a completion in the SMOD system, but note that a completion in the SRPT- $k$  system is equivalent.

- If the completion happens in the system with fewer jobs, assuming this is the SMOD system, we have

$$\begin{array}{lll} b^{SMOD}(t^{(-)}) : & \dots & 0, \quad \delta, \quad p_{i+2}, \quad \dots \\ b^{SRPT-k}(t^{(-)}) : & \dots & q_i, \quad q_{i+1}, \quad q_{i+2}, \quad \dots \end{array} ; \quad \begin{array}{lll} b^{SMOD}(t^{(+)}) : & \dots & 0, \quad 0, \quad p_{i+2}, \quad \dots \\ b^{SRPT-k}(t^{(+)}) : & \dots & q_i, \quad q_{i+1}, \quad q_{i+2}, \quad \dots \end{array} .$$

We again see a 0 appear where the near-completion job was, causing no change to any indices or  $r^+$ . Again, if SMOD and SRPT- $k$  are swapped, nothing changes.

- Finally, if a completion occurs in the system with more jobs, the only change as a result of the completion is the removal of the first index. This index induces no difference at the time of the completion. All other pairs of jobs remain paired and do not change their differences. Thus,  $r^+$  does not change as a result of the completion.

**Continuous change due to work being completed:** We again consider three cases: Equal number of jobs in both systems, more jobs in the SRPT- $k$  system, and more jobs in the SMOD system.

- $N^{SRPT-k} = N^{SMOD}$ : If both systems have the same number of jobs, then both policies will serve the same set of indices: The  $k$  jobs of least remaining size, or all of the jobs if there is less than  $k$ . No differences will change, and the positive part  $r^+$  is constant.
- $N^{SRPT-k} > N^{SMOD}$ : If there are more jobs in the SRPT- $k$  system, both policies serve the  $k$  jobs of least remaining size. In particular, every index being served in the SRPT- $k$  system corresponds to an index that is being either served in the SMOD system or is an empty padding index. In the former case, the difference does not change. In the latter case, the

difference is negative, and has zero positive part. The only way  $r^+$  can change is due to the other indices that are served in the SMOD system but not in the SRPT- $k$  system, and this decreases  $r^+$ .

- $N^{SRPT-k} < N^{SMOD}$ : In this case, we closely look at the *eligible jobs* for the SMOD policy. From Definition 3.3, recall that the  $k + d$  jobs with the smallest size are eligible for service (where  $d = N^{SMOD} - N^{SRPT-k}$ ). Among these  $k + d$  jobs, SMOD gives high priority to jobs with a remaining size of at least the corresponding SRPT- $k$  job, which we referred to as *zero-or-positive difference indices*. Therefore, the set of *eligible jobs* contains all jobs at indices that are either being served in the SRPT- $k$  system, or are empty padding indices.

Let us condition on the number of indices of eligible jobs that have zero or positive difference  $b_i^{SMOD} - b_i^{SRPT-k}$ . The rate of change of  $r^+$  can be characterized based on the number of such indices being served in each system:

- For each index with zero or positive difference that both SMOD and SRPT- $k$  serve,  $r^+$  decreases at a rate of  $1/k$ .
- For each index that SRPT- $k$  serves (with zero or positive difference),  $r^+$  increases at a rate of  $1/k$ .
- All other services do not affect  $r^+$ : Service to negative-difference indices, or service by SMOD to zero-difference indices which are not being served by SRPT- $k$ .

Note that SMOD avoids ever serving indices in this last group (zero-difference indices served by SMOD but not SRPT- $k$ ). From Definition 3.3, SMOD only serves indices that are either padding indices (which always have positive difference) or that are also being served in the SRPT- $k$  system. This is a key property of the SMOD policy.

Thus, to prove that  $r^+$  never increases, it suffices to show that the number of zero-or-positive-difference indices served by SMOD is at least the number of nonnegative indices served by SRPT- $k$ .

If there are at least  $k$  nonnegative indices within the eligible set, SMOD serves exactly  $k$  such indices, as it prioritizes zero-or-positive difference indices over all negative-difference indices. This corresponds to at least as many zero-or-positive difference indices as SRPT- $k$  because only  $k$  jobs can be served simultaneously.

If there are fewer than  $k$  such indices within the eligible set, SMOD serves all of them. Because SRPT- $k$  only serves jobs that are within the eligible set, SMOD again serves at least as many zero-or-positive difference indices as SRPT- $k$ .

This completes the proof. □

## E Details of proof of Lemma 6.5

To finalize the proof, we show that zig-zag matching is preserved under arrivals, completions, and continuous service.

**PROOF.** We analyze the effect of each of the three events in the zig-zag matching property.

**Jump change on arrivals:** Let  $x$  be the size of the arriving job,  $i$  the index at which it is inserted into the SMOD system's  $r$  vector at time  $t$ , and  $t^{(-)}$  the left limit as we approach time  $t$ , as in the proof of Lemma 6.2.

We have  $x \leq b_i^{SMOD}(t^{(-)})$ , as  $x$  is inserted between  $b_{i-1}^{SMOD}(t^{(-)})$  and  $b_i^{SMOD}(t^{(-)})$ . By the zig-zag property, we thus have  $x \leq b_{i+1}^{SRPT-k}(t^{(-)})$ . Thus,  $x$  is inserted at index at most  $i + 1$  in the SRPT- $k$  system, at most 1 position after its location in the SMOD system.

Let's split into two cases: When  $x$  is inserted at index  $i + 1$  in the SRPT- $k$  system, and when it is inserted at an index  $\leq i$ .

- In the first case, the two insertion locations are matched with each other, and no matchings change, so zig-zag matching is preserved.
- In the second case, let  $j$  be the insertion location in the SRPT- $k$  system. Notice that all jobs in the SMOD system up to index  $i$  have remaining size at most  $x$ , while all jobs in the SRPT system at or above index  $j$  have value at least  $x$ . Thus all matchings from  $b_{j-1}^{SMOD}$  to  $b_i^{SMOD}$  obey the zig-zag-matching property. Note also that no other matchings change.

Thus, arrivals maintain the zig-zag property.

**Jump change in service:** If a completion occurs in the system with equal or more jobs, the index will become a padding index. Because the systems change continuously, the property is preserved. Otherwise, there is a padding index present in the other system, resulting in a pair of zeros being removed, which does not affect the zig-zag matching property.

**Continuous change in service:** There are three cases: Equal jobs in both systems, fewer jobs in the SMOD system, or more jobs in the SMOD system.

- If there are equal numbers of jobs in both systems, then both systems serve the first  $k$  indices. This decreases  $b_i^{SMOD}$  and  $b_i^{SRPT-k}$  at rate  $1/k$  for all  $i \leq k$ . In particular, this preserves the zig-zag matching – each zig-zag difference  $b_{i+1}^{SRPT-k} - b_i^{SMOD}$  is unchanged for  $i < k$ , and can only increase for  $i = k$ .
- If there are fewer jobs in the SMOD system, then every job served by SRPT- $k$  is either zig-zag matched with a job that is served by SMOD, or with a padding index. In particular, SMOD is serving at least up to index  $k+1$ , while SRPT- $k$  is serving at most index  $k$ . Thus, the inequality is preserved.
- If there are more jobs in the SMOD system, there can only be one additional job, as proven above. Given that SMOD has exactly one additional job, the SMOD eligible set consists of the  $k+1$  jobs of least remaining size in the SMOD system, and SMOD serves all but 1 such job. There are two cases to consider: All of the indices in the eligible set have nonnegative difference, or there is a negative difference index.
  - In the former case, SMOD will serve the  $k$  jobs of least remaining size, as that is its tiebreaker. In particular, SMOD serves indices 1 through  $k$ , while SRPT- $k$  serves indices 2 through  $k+1$  due to its one padding index. All service is zig-zag matched, and no inequalities are violated.
  - In the latter case, SMOD will omit a job with a negative index from service. Let's denote it  $i$ . We have  $b_i^{SMOD} < b_i^{SRPT-k}$ , because of the negative difference, and  $b_i^{SRPT-k} \leq b_{i+1}^{SRPT-k}$ , due to the ordering of the indices. Thus, we have  $b_i^{SMOD} < b_{i+1}^{SRPT-k}$ . In other words, the zig-zag inequality is strict in the index that SMOD omits from service, and by the time it would become nonstrict (an equality), SMOD will change its service.
For all other indices that are zig-zag matched with an SRPT- $k$  index (all indices  $\leq k$ ), SMOD serves those indices, maintaining the inequalities.

Thus, zig-zag matching is preserved by the continuous change of service.

This completes the proof. □

## F Details of Proof of Lemma 6.12

To finalize the proof, we only need to show that PLN is preserved.

**PROOF.** We analyze the effect on the PLN property of each of the following events in the system: arrival, continuous service, completion, and a difference reaching zero.

**Arrival:** Let the arriving job have size  $x$ ,  $i$  be the index where the job is inserted in the SMOD system, and  $j$  the corresponding index in the SRPT- $k$ . Let the arrival be at time  $t$ . For ease of

notation (similarly to Appendix D), let  $p$  denote the SMOD vector  $b^{SMOD}(t^{(-)})$  just before time  $t$ , and  $q$  the SRPT- $k$  vector  $b^{SRPT}(t^{(-)})$ . We study the cases  $i = j$ ,  $i < j$  and  $i > j$  separately.

- If  $i = j$ , the new job is placed at the same index in both systems. Then, the sequence of differences only changes by the insertion of a new zero-diff entry at index  $i$ . In particular, the relative ordering of positive and negative diff indices is unchanged, preserving PLN.
- If  $i < j$ , we see the following changes in  $p$  and  $q$  at time  $t$ :

$$\begin{array}{ll} b^{SMOD}(t^{(-)}) : & p_i, \quad p_{i+1}, \quad \cdots \quad p_{j-1}; \quad b^{SMOD}(t^{(+)}) : & x \quad p_i, \quad \cdots \quad p_{j-2}, \quad p_{j-1} \\ b^{SRPT-k}(t^{(-)}) : & q_i, \quad q_{i+1}, \quad \cdots \quad q_{j-1}; \quad b^{SRPT-k}(t^{(+)}) : & q_i, \quad q_{i+1}, \quad \cdots \quad q_{j-1}, \quad x \end{array}$$

Note that the only diffs that are affected are indices in the interval  $[i, j]$  at time  $t^{(+)}$ . We thus focus on PLN relationships involving these indices. For all  $\ell \geq i$ , we have  $p_\ell \geq x$ , and for all  $m < j$ ,  $q_m \leq x$ . Thus, all indices in the interval  $[i, j - 1]$  were zero or negative-diff indices at time  $t^{(-)}$ .

If all such indices were zero-diff, then all of the indices involved must have remaining size exactly  $x$ :  $p_i = p_{i+1} = \dots = p_{j-1} = q_i = \dots = q_{j-1} = x$ . In this case, all indices in the interval  $[i, j]$  are zero-diff at time  $t^{(+)}$ , maintaining PLN regardless of which other indices are positive or negative.

Now, we turn to the case where at least one index in the interval  $[i, j - 1]$  was negative-diff at time  $t^{(-)}$ . Because the PLN property holds at time  $t^{(-)}$ , all positive-diff indices at time  $t^{(-)}$  are below  $i$ . Then, only zero-diff and negative-diff indices are created in the interval  $[i, j]$  at time  $t^{(+)}$ , preserving PLN.

- If  $i > j$ , the same argument similar to the case  $i < j$ . Either all indices in the interval  $[i, j - 1]$  at time  $t^{(-)}$  were zero-diff and equal to  $x$ , or there was a positive-diff index. In the first case, PLN is preserved regardless of which other indices are positive or negative. In the latter case, because PLN held at time  $t^{(-)}$ , any negative-diff indices at time  $t^{(-)}$  are at indices  $j$  or greater. Then, they create new positive-diff indices at time  $t^{(+)}$  in the interval  $[i, j]$  and PLN is preserved.

**Continuous service:** If  $N^{SMOD} = N^{SRPT-k}$ , then both systems serve the same set of indices, and the difference  $b_j^{SRPT-k} - b_i^{SMOD}$  is constant for all indices  $i$ . Hence, PLN is maintained.

If  $N^{SMOD} = N^{SRPT-k} + 1$ , we must examine the first  $k + 1$  indices to determine SMOD's service. Within this case, if  $N^{SRPT-k} < k$ , all indices are served in the SMOD system. The only index for which the diff  $b_i^{SRPT-k} - b_i^{SMOD}$  changes is the index  $i = 1$ . Index  $i = 1$  is a positive diff index, because  $b_1^{SRPT-k} > 0$  and  $b_1^{SMOD} = 0$ . As service occurs, the differences towards 0, which does not affect PLN. No other diffs change, so PLN is maintained.

Otherwise, within the case  $N^{SMOD} = N^{SRPT} + 1$ , there are two possibilities, depending on whether there is a negative-diff index among the first  $k + 1$  indices.

- If there is no negative diff among the first  $k + 1$  indices, SMOD serves the first  $k$  indices. SRPT- $k$ , instead, serves the  $k + 1$  index, so the diff of index  $k + 1$  increases at rate  $1/k$ . If index  $k + 1$  previously had zero diff, this may create a new positive-diff index.

Regardless of whether a new positive-diff index is created, this service maintains the PLN property because all negative-diff indices are larger than the  $k + 1$  index, by the assumption of this case.

The only other index where service differs between the two systems is index 1, where SMOD has a padding zero. This is a positive-diff index, so that service also maintains PLN. All other indices receive the same service in both systems, maintaining the diffs and PLN.

- If there is a negative diff among the first  $k + 1$  indices, SMOD omits the largest negative-diff index from service, and serves the rest of the first  $k + 1$  indices.

SMOD's strategy to omit one such index from service causes the negative-diff of that index to increase towards 0. This maintains PLN, as only creating new nonzero diffs can affect PLN. All other indices maintain PLN for the same reasons as in the first case. Note that when the diff reaches zero, the scheduling policy changes. This corresponds to a completion, which we handle later in the proof.

**Completions:** Completions do not change the ordering of positive, zero, or negative differences. In particular, a completion may lead to a padding index being created or removed. Creating a padding index when a job completes has no effect on the differences, as the remaining size of the index continuously changes to 0. Removing a padding index shifts all remaining indices down by one when a job completes in the system with fewer jobs present. However, it does not change the order of the differences. In particular, it always maintains the PLN property.

**A difference reaches zero:** At the moment that indices reach zero difference, the SMOD service option changes discontinuously. However, the PLN property is always maintained if a positive or negative-diff index becomes a zero-diff index, so these discontinuities maintain the PLN property.

Thus, in all cases, PLN is preserved, as desired.  $\square$

## G Details or proof of Lemma 6.14

Similarly to the approach in the previous proofs, we analyzed in detail the effect of each event.

**PROOF.** We analyze the effects of the following events: arrival, continuous service, completion, and a difference reaching zero.

**Arrival:** As discussed in Lemma 6.12, arrivals can only create nonzero-diff indices if the size of the arriving job is no more than the size of a job at a nonzero-diff index in one of the two systems. As  $i_*(t_2)$  is the largest nonzero-diff index, and it is a negative-diff index where  $b_{i_*(t_2)}^{SRPT-k} > b_{i_*(t_2)}^{SMOD}$ , the arriving job must have size smaller than  $b_{i_*(t_2)}^{SRPT}$ . As a result, the arriving job is inserted at an index below both  $i_*(t_2^-)$  and  $m_{t_1}(t_2^-)$ , increasing both indices by one and maintaining the property  $i_*(t_2) \leq m_{t_1}(t_2)$ .

**Continuous service:** We follow the same approach as in the proof of Lemma 6.12 and split into two cases:  $N^{SMOD}(t_2) = N^{SRPT-k}(t_2)$  or  $N^{SMOD}(t_2) = N^{SRPT-k}(t_2) + 1$ .

- If  $N^{SMOD}(t_2) = N^{SRPT-k}(t_2)$ , the same indices are served in both systems, so no differences change. Hence,  $i_*(t_2)$  and  $m_{t_1}(t_2)$  do not change, so the property is maintained.
- If  $N^{SMOD}(t_2) = N^{SRPT-k}(t_2) + 1$ , we must verify that  $i_*(t_2)$  does not increase. Note that no new index with nonzero difference is created. If  $i_*(t_2) > k + 1$ , then no service occurs to indexes at or above  $i_*(t_2)$  and their differences do not change. Otherwise, the  $i_*(t_2)$  index is served in the SRPT- $k$  system. In this case, the  $i_*(t_2)$  index is omitted from service by SMOD, because it must be the largest negative-diff index among those  $k + 1$  indices that are eligible for service. Thus, all indices above  $i_*(t_2)$  up to  $k + 1$  are served by SMOD. As these indices are served in both systems, their differences do not change and  $i_*(t_2)$  does not increase, as desired. Hence, the property is preserved.

**Completion:** A completion affects the system similarly: a new nonzero-diff index cannot be created, so if a padding index is removed in both systems, both  $i_*(t_2^-)$  and  $m_{t_1}(t_2^-)$  decrease by 1, and otherwise  $i_*(t_2^-)$  and  $m_{t_1}(t_2^-)$  are unchanged. In either case,  $i_*(t_2) \leq m_{t_1}(t_2)$  is maintained.

**A difference reaches zero:** When a (previously) nonzero difference reaches zero,  $i_*(t_2)$  may decrease. If  $i_*(t_2^-)$  was the only nonzero-diff index, then the system states have become identical

and SMOD dominates SRPT- $k$ , which persists until  $t_{comp}$ . But  $i_*(t_2)$  cannot increase, because no new nonzero-diff index was created. Thus,  $i_*(t_2) \leq m_{t_1}(t_2)$  is maintained.

We find that in all cases, the  $i_*(t_2) \leq m_{t_1}(t_2)$  property is maintained until  $t_{comp}$  or until SMOD dominates SRPT- $k$ , so at time  $t_{comp}$  SMOD dominates SRPT- $k$ .  $\square$