On the Effect of Inexact Size Information in Size Based Policies

Adam Wierman[1]

1 Introduction

Recently, there have been a number of scheduling success stories in computer applications. Across a wide array of applications, the simple heuristic of "prioritizing small jobs" has been used to reduce user response times with enormous success. As a result of the attention given to size based policies by computer systems researchers, there has been a resurgence in analytical work studying these policies. However, the policies studied in theory, e.g. Shortest-Remaining-Processing-Time (SRPT) and Preemptive-Shortest-Job-First (PSJF), are idealized versions of the policies implemented by practitioners. Thus, there exists a gap between the results provided by theoretical research and the needs of practitioners. This gap results from three primary disconnects between the model studied in theory and the needs of system designers. First, in designing systems, the goal is not simply to provide small response times; other performance measures are also important. Thus, idealized policies such as SRPT and PSJF are often tweaked by practitioners to perform well on secondary performance measures (e.g. fairness and slowdown) [2, 6, 7]. Second, the overhead involved in distinguishing between an infinite number of different priority classes typically causes system designers to discretize policies such as SRPT and PSJF so that they use only a small number of priority classes (5-10) [3, 6]. Third, in many cases information about the service demands (sizes) of jobs is inexact. For instance, when serving static content, web servers have exact knowledge of the sizes of the files being served, but have inexact knowledge of network conditions. Thus, the web server only has an estimate of the true service demand [5, 7].

An emerging style of research attempts to bridge the gap between theoreticians and practitioners by studying *classifications of scheduling policies* instead of individual idealized policies. The goal is to formalize a scheduling heuristic such as "prioritizing small jobs" and then study the impact of this *heuristic* instead of studying one specific policy that obeys the heuristic. One such heuristic classification is the SMART class, introduced in [9], which formalizes the heuristic of "prioritizing small jobs." The SMART classification includes policies such as SRPT and PSJF in addition to a wide array of other static and dynamic priority policies. Thus, the SMART class bridges the first of the disconnects we discussed above: it includes hybrid policies that are used in practice in order to perform well for both response time and secondary metrics. However, the SMART class does not help to bridge the other two disconnects above. In this work, we present a generalization of the SMART class, the SMART_ϵ class, which attempts to bridge all three disconnects above. In particular, our main result (Theorem 2) provides a simple characterization of the tradeoff between the accuracy of job size estimates and the mean response times of policies that prioritize small jobs.

2 Defining the SMART_ϵ class

We define the class of SMART_ϵ policies as follows. In the definition, jobs will be denoted by a, b, or c. Job a will have remaining size r_a, original size s_a, and arrival time t_a. The original sizes, remaining sizes, and arrival times of b and c are defined similarly. Throughout, we define *job a to have priority over job b* if job b can never run while job a is in the system.

Definition 1 *For all $x \in [0, \infty)$, define $x_\epsilon = \epsilon(x) \geq x$ for some function ϵ. A work conserving policy $P \in \text{SMART}_\epsilon$ if it obeys the following properties at all times.*

- **Weak Bias Property:** *If $s_a = x$ and $r_b > x_\epsilon$, then job a has priority over job b.*
- **Consistency Property:** *If job a ever receives service while job b is in the system, then at all times thereafter job a has priority over job b.*
- **Transitivity Property:** *If an arriving job b preempts job c, then thereafter, until job c receives service, every arrival a with size $s_a = x$ such that $x_\epsilon < s_b$ is given priority over job c.*

[1]Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA, acw@cs.cmu.edu

The $\texttt{SMART}_\epsilon$ class is defined in an almost parallel way to the SMART class in [9]. In particular, the SMART class is a subclass of $\texttt{SMART}_\epsilon$ that can be obtained by setting $x_\epsilon = x$. Thus, the parameter x_ϵ provides a formal way to capture the effect of "weakening the bias towards small jobs." If, for example, we think of $\epsilon(x) = x(1+\varepsilon)$, as ε grows the bias towards small jobs sizes required of $\texttt{SMART}_\epsilon$ policies decreases.

As with the definition of the original SMART class, each of the three properties in the definition of $\texttt{SMART}_\epsilon$ formalizes a notion of "smart" scheduling. The Weak Bias Property guarantees that the job being run at the server has remaining size not too much larger than the original size of every job in the system, which formalizes the idea of "prioritizing small jobs." The Consistency and Transitivity Properties ensure coherency in the priority structure enforced by the Bias Property. In particular, the Consistency Property prevents time-sharing by guaranteeing that after job a is chosen to run ahead of b, job b will never run ahead of job a. This makes intuitive sense because all $\texttt{SMART}_\epsilon$ policies are based on the heuristic of giving priority to small jobs, and as job a receives service, it can only get smaller. Finally, the Transitivity Property guarantees that $\texttt{SMART}_\epsilon$ policies do not second guess themselves: if an arrival a is determined to be "smaller" than job b in some sense (i.e. is given priority over job b), future arrivals with smaller size than a should also considered "smaller" than b until b receives service.

3 Examples of SMART$_\epsilon$ policies

Of course, the $\texttt{SMART}_\epsilon$ class includes all SMART policies. Thus, it includes SRPT and PSJF in addition to a wide range of hybrids of these policies. For example, SMART includes any static priority policy that gives each job of size s and remaining size r a priority using a fixed priority $p(s,r)$ such that for $s_1 \leq s_2$ and $r_1 < r_2$, $p(s_1, r_1) < p(s_2, r_2)$ schedules the job with the lowest priority. Thus, examples of hybrid policies in SMART include $p(s,r) = s^i r^j$ for all $i \geq 0$ and $j > 0$. Interestingly, many such examples improve on SRPT and PSJF with respect to weighted response time measures such as mean slowdown. In addition, SMART also includes *time-varying policies*, i.e. policies that can change their priority rules over time based on system-state information or randomization. These generalizations are possible because the SMART definition enforces only a *partial ordering* on priorities of jobs in the system. It is of enormous practical importance that time-varying policies are included in SMART because it allows system designers to use the class in order to perform online multi-objective optimization. Specifically, suppose a system designer wants to optimize a secondary objective while still providing small mean response times. In order to accomplish this, the system designer can implement a parameterized version of SMART, such as prioritizing based on $p(s,r) = s^i r^j$, and then use machine learning techniques to search the space (i,j) online for the SMART policy that optimizes the secondary objective. This technique can be extremely useful in web applications where the service distribution is time-varying and thus the optimal scheduling policy often is not static.

In addition to SMART policies, $\texttt{SMART}_\epsilon$ includes many practical policies that are excluded from SMART because of the rigidity of the original Bias Property. First, notice that $\texttt{SMART}_\epsilon$ can include policies that have only a finite number of priority levels. In particular, it can include "threshold based policies" where there are a finite number of thresholds $0 = a_1, \ldots, a_n = \infty$ and a job of size x is assigned priority $p(s,r) = i$ if $p'(s,r) \in [a_i, a_{i+1})$ for some static priority function $p'(s,r) \in$ SMART such as $p'(s,r) = s$ (i.e. PSJF). As we mentioned earlier, it is of particular practical importance to include these policies because in many cases system designers simplify implementations by using only 5-10 priority levels.

In addition to including threshold based policies, $\texttt{SMART}_\epsilon$ includes SMART policies that are being run using inexact job size information. When one performs SRPT or some other SMART policy on job size estimates, the resulting policy is not in SMART and is difficult to study directly; however the resulting situation does fall into $\texttt{SMART}_\epsilon$ for a suitable $\epsilon(x)$. As an example, if the inexact job sizes are a result of a time-varying service capacity (such as a web server sharing bandwidth with other hosts), taking $\epsilon(x) = (1+\varepsilon)x$ models the situation where the maximum change in service rate is ε.

4 Bounding response times under SMART$_\epsilon$

We now present bounds on the response times of policies in the SMART$_\epsilon$ class. Before stating the results, we need to introduce some notation. Throughout we will consider an M/GI/1 system with a continuous service distribution having finite mean. Let $\rho < 1$ be the system load. That is $\rho = \lambda E[X]$, where λ is the arrival rate of the system and X is a random variable distributed according to the service (job size) distribution $F(x)$ having density function $f(x)$. Let $\overline{F}(x) = 1 - F(x)$. The response time for a job of size x under scheduling policy P is $T(x)^P$, and the overall response time under scheduling policy P is T^P, where $E[T]^P = \int_0^\infty E[T(x)]^P f(x)dx$. Further, define $m_i(x) = \int_0^x t^i f(t)dt$ and $\rho(x) = \lambda m_1(x)$. Let $B_x(Y)$ be a random variable distributed as the length of a M/GI/1 busy period with arrival rate λ and service distribution $XI(X < x)$ started by an amount of work Y. Finally, let Q_x^{SRPT} be the stationary amount of work in an M/GI/1 SRPT queue contributed by jobs having remaining size less than x.

Theorem 1 *In an M/GI/1 queue, for all $P \in$ SMART$_\epsilon$,*

$$T(x)^P \leq_{st} B_{x_\epsilon}(x + Q_{x_\epsilon}^{SRPT}).$$

Thus,

$$E[T(x)]^P \leq \frac{x}{1-\rho(x_\epsilon)} + \frac{\lambda m_2(x_\epsilon) + \lambda x_\epsilon^2 \overline{F}(x_\epsilon)}{2(1-\rho(x_\epsilon))^2}$$

Though the behavior of $T(x)$ is not the primary metric of interest in most situations, using Theorem 1 we can obtain the following bound on the overall mean response time under SMART$_\epsilon$ policies. Recall that $E[T]^{SRPT}$ is optimal [8], so we will define $E[T]^{OPT} = E[T]^{SRPT}$.

Theorem 2 *Consider an M/GI/1 queue with $P \in$ SMART$_\epsilon$. If there exists an $\varepsilon \geq 0$ such that $x_\epsilon \leq (1+\varepsilon)x$ for all x and a $0 \leq \gamma \leq 1$ such that $\rho(x_\epsilon) - \rho(x) \leq \gamma(\rho - \rho(x))$, then*

$$E[T]^P \leq 2\left(\frac{1+\varepsilon}{1-\gamma}\right)^2 E[T]^{OPT}$$

Theorem 2 illustrates the tradeoff between the rigidity of the bias towards small jobs in a policy (as characterized by x_ϵ) and the mean response time provided by the policy. Notice that the strongest bias towards small job sizes corresponds to $x_\epsilon = x$ (i.e. $\gamma = \epsilon = 0$), in which case we obtain a bound on the behavior of SMART policies that matches the bound proven in [9]. As x_ϵ grows, Theorem 2 characterizes how the mean response time of policies grows using two conditions that present complementary formulations of the price paid for weakening the Bias Property: you can let significantly larger jobs have priority without paying a price in mean response time as long as the larger jobs do not make up too much load. Clearly, the tradeoff between these two conditions will vary depending on the service distribution.

One particularly important service distribution under which to study the tradeoff is the Pareto distribution, which is often suggested as a good model of service distributions for web applications. Under a Pareto distribution $\overline{F}(x) = (\beta/x)^\alpha$. In this case, the two conditions in Theorem 2 can be simplified:

Corollary 3 *Consider an M/GI/1 queue with $P \in$ SMART$_\epsilon$ and $X \sim$ Pareto(α, β). If there exists an $\varepsilon \geq 0$ such that $x_\epsilon \leq (1+\varepsilon)x$ then*

$$E[T]^P \leq 2(1+\varepsilon)^{2\alpha} E[T]^{OPT}.$$

In addition to characterizing the effect of the rigidity of the bias towards small jobs in a policy, Theorem 2 and Corollary 3 can also be used to characterize the effect of inexact knowledge of job size information. Most prior work in this regard has used simulation experiments, e.g. [5], however SMART$_\epsilon$ allows easy back-of-the-envelope calculations. For example, if we have a Pareto service distribution with $\alpha = 1.1$ and job sizes can be estimated to within a factor of 50%, any SMART$_\epsilon$ policy (which includes running SRPT or PSJF on the job size estimates) still provides response times within a factor of 4.88 of the optimal regardless of the load. This factor is quite small given that the variance is infinite and the bound holds for ρ arbitrarily close to 1. As a comparison no finite constant bound is possible for many common policies, including First-Come-First-Served and Processor-Sharing.

Finally, we can also view Theorem 2 and Corollary 3 as a strong statement about the behavior of policies with a finite number of priority classes. Though there has been a lot of work analyzing such policies [1, 4], it is still difficult to understand how far from optimal the mean response times of these policies are due to the fact that determining the optimal threshold values is typically not tractable when the policy is preemptive (see [1] for a discussion). However, $\mathtt{SMART}_\epsilon$ again provides simple back-of-the-envelope calculations that help understand how close the mean response times of policies with only a finite number of priority classes are to optimal. In particular, the condition of $x_\epsilon < (1+\varepsilon)x$ provides a direct relation between the spacing of thresholds (i.e. the range of job sizes included in each priority level) and the guarantee on mean response time. For example under a Bounded Pareto service distribution defined on $[x_L, x_U]$, we have:

Corollary 4 *Consider an M/GI/1 queue with $X \sim BoundedPareto(\alpha, \beta, x_L, x_U)$. If P is a threshold based policy with thresholds $x_L = a_0, \ldots, a_n = x_U$ such that $a_i \le (1+\varepsilon)a_{i-1}$ for $i = 1, \ldots, n$ then*

$$E[T]^P \le 2(1+\varepsilon)^{2\alpha} E[T]^{OPT}.$$

Further, if $a_i = x_L(1+\varepsilon)^i$ and $n = \min(i \in \mathbb{N} : a_i \ge x_U)$ then

$$E[T]^P \le 2\left(\frac{x_U}{x_L}\right)^{2\alpha/n} E[T]^{OPT}.$$

There are a few interesting observations about Corollary 4. First, notice the simplicity with which one can understand the performance gain obtainable by adding priority classes. For example, if $\alpha = 1.1$, $x_L = 300$, and $x_U = 10^9$ then we have that using $n = 15$ priority classes guarantees that mean response times are within a factor of 18.10 of optimal, *even for ρ arbitrarily close to 1.* If we instead use $n = 25$ priority classes the factor drops to 7.50. Further, if we let $n \to \infty$ the factor converges to 2, which is the bound on $\mathtt{SMART}$ policies in [9]. Second, it is interesting to point out the importance the finite upper bound on the service distribution plays in Corollary 4. If the service distribution were unbounded, it would require an infinite number of thresholds for a threshold based policy to satisfy Theorem 2. This may seem like a restriction, but it is possible to show that in an M/GI/1 queue with an unbounded Pareto service distribution any threshold based policy P with a finite number of thresholds will have

$$\lim_{\rho \to 1} E[T]^P / E[T]^{OPT} = \infty.$$

Thus, some restriction is necessary.

References

[1] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling.* Addison-Wesley Publishing Company, 1967.

[2] M. Gong and C. Williamson. Simulation evaluation of hybrid SRPT scheduling policies. In *Proc of IEEE MASCOTS*, 2004.

[3] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Implementation of SRPT scheduling in web servers. *ACM Transactions on Computer Systems*, 21(2), May 2003.

[4] L. Kleinrock. *Queueing Systems*, volume II. Computer Applications. John Wiley & Sons, 1976.

[5] D. Lu, H. Sheng, and P. Dinda. Size-based scheduling policies with inaccurate scheduling information. In *Proc. of IEEE Mascots*, 2004.

[6] I. A. Rai, G. Urvoy-Keller, M. Vernon, and E. W. Biersack. Performance modeling of LAS based scheduling in packet switched networks. In *Proc. of ACM Sigmetrics-Performance*, 2004.

[7] M. Rawat and A. Kshemkalyani. SWIFT: Scheduling in web servers for fast response time. In *Symp. on Net. Comp. and App.*, 2003.

[8] L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678–690, 1968.

[9] A. Wierman, M. Harchol-Balter, and T. Osogami. Nearly insensitive bounds on SMART scheduling. In *Proc. of ACM Sigmetrics*, 2005.