Self-Adaptive Admission Control Policies for Resource-Sharing Systems

Varun Gupta
Carnegie Mellon University
varun@cs.cmu.edu

Mor Harchol-Balter *
Carnegie Mellon University
harchol@cs.cmu.edu

ABSTRACT

We consider the problem of admission control in resource sharing systems, such as web servers and transaction processing systems, when the job size distribution has high variability, with the aim of minimizing the mean response time. It is well known that in such resource sharing systems, as the number of tasks concurrently sharing the resource is increased, the server throughput initially increases, due to more efficient utilization of resources, but starts falling beyond a certain point, due to resource contention and thrashing. Most admission control mechanisms solve this problem by imposing a fixed upper bound on the number of concurrent transactions allowed into the system, called the Multi-Programming-Limit (MPL), and making the arrivals which find the server full queue up. Almost always, the MPL is chosen to be the point that maximizes server efficiency.

In this paper we abstract such resource sharing systems as a Processor Sharing (PS) server with state-dependent service rate and a First-Come-First-Served (FCFS) queue, and we analyze the performance of this model from a queueing theoretic perspective. We start by showing that, counter to the common wisdom, the peak efficiency point is not always optimal for minimizing the mean response time. Instead, significant performance gains can be obtained by running the system at less than the peak efficiency. We provide a simple expression for the static MPL that achieves near-optimal mean response time for general distributions.

Next we present two traffic-oblivious dynamic admission control policies that adjust the MPL based on the instantaneous queue length while also taking into account the variability of the job size distribution. The structure of our admission control policies is a mixture of fluid control when the number of jobs in the system is high, with a stochastic component when the system is near-empty. We show via simulations that our dynamic policies are much more robust to unknown traffic intensities and burstiness in the arrival process than imposing a static MPL.

*Research supported by NSF SMA/PDOS Grant CCR-0615262 and a 2009 IBM Faculty Award.

SIGMETRICS/Performance'09, June 15–19, 2009, Seattle, WA, USA.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance—*modeling and prediction, queueing theory*; G.1.6 [**Numerical Analysis**]: Optimization—*stochastic programming*

General Terms

Performance

1. INTRODUCTION

The notion of time-sharing has been around since the earliest days of operating systems, as described in the first paper on Unix [23]. Time-sharing has several benefits. First, given that jobs often need different resources (CPU, I/O) at different times, time-sharing allows for increased throughput, typically allowing two jobs to complete in the same time as one, since they aren't likely to need the same resources at the same time. Another major benefit of time-sharing is that it allows small jobs to get out quickly; the small jobs are not stuck queueing behind big jobs as they would be in a first-come-first-served (FCFS) system, and therefore they don't have to suffer the delays of waiting for big jobs to complete.

However, as many researchers have observed, time-sharing is most effective when there is a fixed Multi-Programming-Limit (MPL) imposed, so that not too many jobs can time-share at once. Allowing too many jobs to time-share can lead to thrashing (due to the context-switching overhead), and reduced overall performance. This point has been observed time and time again starting with operating systems papers in the 1970's [8] and 1980's [5, 2], and continuing to more recent Web server design papers [9, 13], and database implementation papers [24, 12]. Specifically, a system has a service rate curve which shows that the "speed" of the system increases when the number of jobs in the system increases from 1 to 2, and increases again as the number increases from 2 to 3, but the system speed starts to drop as the number of jobs in the system increases beyond some point. Figure 1 shows a typical service rate curve (see, e.g. [26, Figure 2]).

Model

To model a time-sharing system, we start with a G/G/1/PS queue where PS denotes "processor sharing," meaning that if there are n jobs in the system, they each receive $\frac{1}{n}$th of the system's processing capacity. We will assume that the job sizes (or service requirements) are independently and identically drawn from a general distribution, and we will

Figure 1: A prototypical service rate curve. The peak efficiency point for the curve shown is $K^* = 5$.

use X to denote such a generic random variable. We will use C^2 to denote the squared coefficient of variation (SCV) of X:

$$C^2 = \frac{var(X)}{\mathbf{E}[X]^2}$$

and throughout assume that $\mathbf{E}[X] = 1$ without loss of generality. In order to capture the fact that the speed of the system depends on the number of jobs at the server, we assume that our $G/G/1/PS$ server has state-dependent service rates $\mu(n)$. That is, when the number of jobs at the server is n, the speed of the server is $\mu(n)$, where $\mu(n)$ is chosen to match the system's service rate curve (Figure 1). As an example, a job of size x seconds which is sharing the server with n jobs (including itself) for its entire duration would require $\frac{x}{\mu(n)} \cdot n$ time to complete. We assume that the $\mu(n)$ curve is unimodal, that is, initially it is non-decreasing and then after some point the curve switches to being non-increasing. We define K' to be the smallest MPL which achieves the maximum speed, and K^* to be the largest MPL which achieves the maximum speed. For the $\mu(n)$ curve in Figure 1, $K' = 4$ and $K^* = 5$.

To complete our model, we now add an MPL parameter which limits the number of jobs that are allowed to concurrently share the server to some number MPL=K, and forces all remaining jobs to wait in a First-Come-First-Served (FCFS) buffer. We assume that the job sizes of the jobs in the system are not known, and size-based prioritization is not possible. We denote our model by the notation G/G/PS-MPL. Figure 2 depicts a G/G/PS-MPL system with MPL=4. When we additionally assume the arrival process to be Poisson, we will denote the system by M/G/PS-MPL. Throughout, we assume load-dependent service rates $\mu(n)$. So, for example, if there are $n = 10$ jobs in the G/G/PS-4 system, the server speed will be $\mu(4)$, since only 4 jobs time-share the server, while if there are $n<4$ jobs in the system, the speed will be $\mu(n)$. Thus the response time for a job of size x will be its queueing time plus its service time, where the service time will typically be $\frac{x}{\mu(4)} \cdot 4$, assuming that there are at least 4 jobs in the system during the job's time in system.

The interesting question for the G/G/PS-MPL model is, of course:

> *What is the optimal MPL, so as to minimize mean response time?*

Obviously, the service rate curve plays a large role in the answer. In fact, computer systems papers would have us believe that the service rate curve provides the entire answer to this question: Simply choose that MPL that maximizes efficiency, e.g., [1, 5, 9]. For the curve shown in Figure 1, this would mean choosing the MPL to be $K' = 4$ or $K^* = 5$. In this paper, we will show that this obvious answer can be far from correct, when job size variability is high. We will also ask and answer the even harder question of how to dynamically vary the MPL when the arrival rate is not known and as load conditions change.

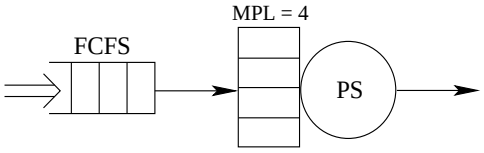


Figure 2: A G/G/PS-MPL queue with $MPL = 4$. Only 4 jobs can simultaneously share the server. The rest must wait outside in FCFS order.

When the job size distribution is Exponential, the answer is straightforward: We always want to operate at the peak efficiency point, regardless of the arrival process. The question of choosing an optimal MPL becomes interesting when the job size distribution exhibits high variability, since with high variability job size distributions, it is known that PS has a much better performance than FCFS because time sharing prevents small jobs from getting blocked behind big jobs. However, by letting too many jobs into the server, the system efficiency drops.

Prior Work

Unfortunately, the question of choosing an optimal MPL for high variability job size distributions is difficult to answer since there is no known analysis even under a Poisson arrival process with a fixed arrival rate. This is not surprising because the M/G/PS-MPL model is a generalization of the classical $M/G/K$ multiserver system where there are K identical servers, each of which can process at most one job at a time, and a FCFS buffer where jobs queue up when all the K servers are busy. The $M/G/K$ multiserver system can be modeled by an M/G/PS-MPL system with MPL $= K$, and $\mu(n) = \mu \cdot n$, where μ is the speed of the individual servers. The performance analysis of the $M/G/K$ system is still largely an open problem. While the performance analysis of the Processor-Sharing queue has been well understood for years, and research on the M/G/1/PS queue has been abundant [14, 15, 7, 16, 27, 29, 6], very little is known about the M/G/PS-MPL queue. Most analyses of the M/G/PS-MPL queue do not allow for load-dependent service rates. For example, Itzhak and Halfin [3] derive a 2-moment approximation for the mean response time for the M/G/PS-MPL queue where the service rate is fixed, while Zhang and Zwart [28] derive a heavy-traffic diffusion approximation for M/G/PS-MPL (which they refer to as the Limited Processor Sharing queue) with a fixed service rate. There is one analysis of the M/G/PS-MPL that does involve state-dependent service rates, see Rege & Sengupta [21]. However [21] requires that job sizes are exponentially-distributed while we are focusing on high-variability job size distributions which are more representative of computer workloads. While Fredericks [10] warns that the exponential job size distribution is not a good indicator of performance of the M/G/PS-MPL with high variability, he does not derive an approximation

that allows for higher variability. Finally none of the above theoretical papers have tried to answer the question of how to set the MPL so as to minimize the mean response time.

While there is a large body of work on adaptive load control and admission control in resource-sharing systems, all of the existing work either ignores the crucial point of load-dependent service rates at the server, or the effect of job size variability. Elnikety et al. [9] propose monitoring the load of the server and admitting tasks as long as the resulting load does not exceed the peak efficiency point. Blake [5] also proposes operating at the peak efficiency point, but uses the fraction of jobs waiting in the virtual memory queue as an indicator of thrashing to control the MPL. Kamra et al. [13] model the server as an ideal $M/G/1/PS$ system thereby ignoring the state-dependence of the service rate. They monitor the response time of the departing jobs, and adjust the dropping probability of the arriving requests to achieve target response time for the admitted tasks. Our solutions differ from [13] in that we do not drop requests. Heiss and Wagner [12] propose a feedback mechanism to monitor the effect that changing the MPL has on the performance metric of interest. However, as the authors observe, this requires monitoring at least hundreds of departures before a control decision can be taken. Another drawback of the solution proposed in [12] is that the authors assume the system reaches stationarity after the control decision has been taken. This assumption is hardly justified, and can cause incorrect decisions due to a delay between the time the control action is taken, and the time its effect is observed. Schroeder et al. [24] consider the problem of setting a static MPL in the presence of variable job sizes, but the emphasis of [24] is to find a sufficiently small MPL so that class-based task prioritization can be done in the FCFS queue. Schroeder et al. also develop a feedback based controller based on measuring the throughput and response times, but ignore the state-dependence of service rate. Van der Weij, Bhulai and van der Mei [25] also look at admission control in a PS queue under the assumption that the job size distribution is of phase type and the phases of all the jobs in the system are known. The authors assume a constant $\mu(n)$, and characterize the optimal admission control policy. In contrast, we assume that no information about the job sizes is available and hence size-based prioritization is not possible.

Contributions of this paper

To the best of our knowledge, we are the first to consider the question of controlling the multi-programming limit in a resource-sharing system by taking into account both the service rate curve, and the high variability of the job size distribution. Our paper has two principal contributions:

1. Optimal traffic-aware static policies

We derive the first approximation for mean response time for the M/G/PS-MPL queue with state-dependent service rates, and extend this approximation for GI/G/PS-MPL systems. The approximation enables us to choose the MPL that minimizes mean response time. Via extensive simulation experiments presented both in this paper and in [11], we demonstrate that the optimal MPL setting can be much higher than the peak efficiency point, under job size variability characteristic of computer workloads. In fact, we show examples where the optimal MPL operates the system at 85% of the peak efficiency, while dropping the mean response time by more than 65% [11]. Our results are verified across a variety of job size distributions including Weibull, Pareto and Hyperexponential distributions. We refer to the static policy which uses the optimal static MPL as the Opt-Static policy.

2. Near-optimal traffic-oblivious dynamic policies

The above results assume jobs arrive according to a Poisson process with a known arrival rate and propose the best *static* MPL. However, we are interested in scenarios where the mean arrival rate may not be known, or the arrival process may not even be Poisson, exhibiting burstiness or temporal correlations. Our goal is to design *light-weight* MPL control policies that adapt to the traffic characteristics. By light-weight policies, we mean policies which take decisions based only on the instantaneous number of jobs in the buffer, $Q(t)$, and the instantaneous number of jobs at the server, $K(t)$.

We first consider the setting where the arrival process is known to be Poisson, but with an unknown mean arrival rate. We find that, unsurprisingly, static MPLs are very poor in handling uncertainty in the mean arrival rate. We then propose two light-weight MPL control policies, Light-Approx and Poisson-Approx that robustly handle uncertainty in the mean arrival rate. The ***key idea in our approach*** is that by considering a special class of job size distributions, the 2-phase degenerate hyperexponential distribution, we are able to incorporate the effect of job size variability in our optimization problem, while $(Q(t), K(t))$ remains a Markov process. Thus, the control policies we obtain are a function only of $(Q(t), K(t))$. Via simulations we show that both Light-Approx and Poisson-Approx are robust at adapting to unknown mean arrival rate, resulting in near-optimal mean response time (under 19%) for a wide range of arrival rates when compared to the optimal static MPLs for each arrival rate.

Next, we consider the setting where not only is the mean arrival rate not known, but the arrival process is also bursty. We demonstrate that both Light-Approx and Poisson-Approx are simultaneously robust to unknown mean arrival rate and burstiness of the arrival process, resulting in less than 25% higher mean response time than the mean response time for the optimal traffic-aware static MPL. Surprisingly, we find that if the mean arrival rate is known, a static MPL optimized for a Poisson arrival process with the given mean arrival rate is also near-optimal when the arrival process is bursty with that mean arrival rate (that is, the interarrival times are i.i.d. but not Exponentially distributed). However, burstiness can greatly worsen the performance of static policies when the mean arrival rate is unknown.

Outline

In Section 2, we solve the problem of choosing the optimal static MPL for a general job size distribution under the assumption that the arrival process is Poisson with a known mean arrival rate. In Section 3, we begin by demonstrating that the approach of choosing a single static MPL is fundamentally limited in its ability to handle variability in traffic arrival patterns. In Sections 3.2 and 3.3, we construct our dynamic MPL control policies Light-Approx and Poisson-Approx, respectively. In Section 3.4, we evaluate these dynamic policies with respect to *(i)* robustness to unknown arrival rate, and *(ii)* robustness to burstiness of the arrival process. Finally we compare our traffic-oblivious dynamic policies to the optimal traffic-aware static MPL policy.

2. CHOOSING THE BEST STATIC MPL

Our first goal in this paper is to address the question of *how to optimally set a multi-programming limit* in a resource-sharing system so as to minimize the mean response time (equivalently, minimize the mean number of jobs in the system). We assume that the arrival process is Poisson with a known mean arrival rate, and that the job size distribution is known. In Section 2.1, we present some stochastic monotonicity results for the performance of PS-MPL systems under fairly general job size distributions which motivate the need to appropriately choose the MPL based on the job size distribution. In Section 2.2, we provide a simple approximation for the mean number of jobs in an M/G/PS-MPL system with state-dependent service rate involving only the first two moments of the job size distribution, and demonstrate a job size distribution for which the approximation is, in fact, exact. In Section 2.3, we present the OPT-STATIC policy, which uses our approximation to choose a static MPL based on the mean arrival rate and the first two moments of the job size distribution. Even though our approximation involves only the first two moments of the job size distribution, we show via experiments that it leads to optimal or near-optimal MPL selection for a range of distributions used to model computer workloads.

2.1 Stochastic monotonicity results

Let F be a distribution function for a non-negative random variable X, and f be the corresponding density function.

Definition 1. Distribution F is said to belong to the class DFR (IFR) if the function $h(x) = \frac{f(x)}{1-F(x)}$ is decreasing (increasing).

Definition 2. Distribution F is said to belong to the class DMRL (IMRL) if the function $R(a) = \mathbf{E}[X - a|X \geq a]$ is decreasing (increasing).

The classes IMRL (Increasing Mean Residual Life, also referred to as NWUE for New Worse than Used in Expectation) and DFR (Decreasing Failure Rate) both capture the notion that young jobs (those who have received less service) are more likely to finish earlier than old jobs. The condition DFR is equivalent to saying that the residual life of young jobs is stochastically smaller than the residual life of old jobs, while IMRL is equivalent to saying that the mean residual life of young jobs is smaller than the mean residual life of old jobs.

The following is a corollary of [19, Theorem 1].

PROPOSITION 1. *In a G/G/PS-MPL system with a DFR job size distribution, the number of jobs in the system at any time is a stochastically decreasing function of the MPL K, for $K \leq K^*$. For an IFR distribution, the number of jobs in the system is a stochastically increasing function of the MPL K, for $K \geq K'$.*

A similar proposition can be proven for the mean number of jobs (equivalently mean response time) by relaxing the assumptions on the arrival process and the job size distribution.

PROPOSITION 2. *In an M/G/PS-MPL system with an IMRL job size distribution, the mean number of jobs in the system is a decreasing function of the MPL K, for $K \leq K^*$. For a DMRL distribution, the mean number of jobs in the system is an increasing function of the MPL K, for $K \geq K'$.*

PROOF. From [22, Theorem 3.14], for IMRL distributions, it suffices to prove that for all x, $\bar{V}_x$, which denotes the mean workload in the system due to jobs with attained service less than x, is decreasing in the MPL K for $K \leq K^*$. From the proof of [19, Theorem 1], this is easily seen to hold. The proof for DMRL distributions is analogous. □

Intuitively, when the job size distribution is DFR or IMRL, we prefer to serve young jobs as they are more likely to finish earlier. By choosing an MPL smaller than K^*, we do not gain serving capacity, since K^* achieves the maximum speed, and simultaneously limits the ability of new jobs (which are likely to be small) to enter service. Similarly, for IFR or DMRL job size distributions, we prefer to serve old jobs as they are more likely to finish earlier. By choosing an MPL larger than K', we do not gain aggregate serving capacity, and we simultaneously reduce the capacity available to old jobs, as young jobs are allowed into service. Job size distributions belonging to class DFR and IMRL correspond to distributions which are more variable than the Exponential distribution, and the above results show that there is no benefit in running at an MPL smaller than K^* in this case. However, there might be benefit in operating at an MPL higher than K^*, increasing the chance for small jobs to enter service and finish quickly even while losing aggregate service capacity in the process, as we show next.

2.2 2-moment approximation for *M/G/PS-MPL*

As mentioned earlier, there are no known analytical expressions or approximations for the mean number of jobs in an M/G/PS-MPL system with state-dependent service rate. We now propose a simple approximation for the mean number of jobs in an M/G/PS-MPL system involving only the first two moments of the job size distribution.

PROPOSITION 3. *Let $\mathbf{E}[N]$ denote the mean number of jobs in an M/G/PS-MPL system with arrival rate λ, state-dependent service rate $\mu(n)$ when there are n jobs at the PS server, with MPL=K, and a general job size distribution with mean 1 and SCV C^2. Then,*

$$\mathbf{E}[N] \approx \mathbf{E}\Big[N^S_{Exp}(K)\Big] + \frac{C^2+1}{2}\mathbf{E}\Big[N^Q_{Exp}(K)\Big] \qquad (1)$$

where $\mathbf{E}\Big[N^Q_{Exp}(K)\Big]$ and $\mathbf{E}\big[N^S_{Exp}(K)\big]$, respectively, denote the mean number of jobs in the FCFS Queue and at the PS Server in an M/M/PS-MPL with the same state-dependent service rates as the original M/G/PS-MPL system, with MPL=K and Exponential job size distribution with mean 1. The expressions for $\mathbf{E}\Big[N^Q_{Exp}(K)\Big]$ and $\mathbf{E}\big[N^S_{Exp}(K)\big]$ are given by:

$$\mathbf{E}\Big[N^Q_{Exp}(K)\Big] = \frac{\phi_{K+1}}{1+\sum_{i=1}^{\infty}\phi_i}\left(\frac{1}{1-\frac{\lambda}{\mu(K)}}\right)^2$$

$$\mathbf{E}\Big[N^S_{Exp}(K)\Big] = \frac{\sum_{i=1}^{K} i\cdot\phi_i + K\cdot\sum_{i=K+1}^{\infty}\phi_i}{1+\sum_{i=1}^{\infty}\phi_i}$$

where ϕ_i's are the ratio of the stationary probabilities and the idle probability for an M/M/PS-MPL, and are given by:

$$\phi_i = \begin{cases} \Pi_{j=1}^{i}\frac{\lambda}{\mu(j)} & 1 \leq i \leq K, \\ \phi_K \cdot \left(\frac{\lambda}{\mu(K)}\right)^{i-K} & i > K. \end{cases}$$

Proposition 3 can be seen as a generalization of the Lee and Longton [17] approximation for the mean number of jobs in

an M/G/K system, and agrees with the approximation given by [3] when the service rate is independent of the state. In Proposition 4, we show that approximation (1) is in fact exact for a degenerate hyperexponential distribution, H^*, with mean 1 and squared of coefficient of variation C^2.

Definition 3. A *degenerate hyperexponential distribution* with mean 1 and SCV C^2 is defined by:

$$H^*(C^2) \sim \begin{cases} 0 & \text{with probability } 1-q = \frac{C^2-1}{C^2+1} \\ \text{Exp}\left(\frac{2}{C^2+1}\right) & \text{with probability } q = \frac{2}{C^2+1} \end{cases}$$

where $\text{Exp}(\nu)$ denotes an Exponential random variable with mean $1/\nu$.

PROPOSITION 4. *The mean number of jobs in an $M/H^*(C^2)$/PS-MPL system with arrival rate λ, state-dependent service rate $\mu(n)$ when there are n jobs at the PS server, and MPL=K is given by:*

$$\mathbf{E}\left[N_{H^*(C^2)}(K)\right] = \mathbf{E}\left[N^S_{Exp}(K)\right] + \frac{C^2+1}{2}\mathbf{E}\left[N^Q_{Exp}(K)\right]$$

where $\mathbf{E}\left[N^Q_{Exp}(K)\right]$ and $\mathbf{E}\left[N^S_{Exp}(K)\right]$ are as defined in Proposition 3.

PROOF. We first observe that the $H^*(C^2)$ distribution consists of two classes of jobs, those of size 0 and those belonging to the Exponential branch. The response time and hence the number of jobs belonging to the Exponential class in the $M/H^*(C^2)$/PS-MPL system is not affected by the presence of zero-sized jobs. Therefore, the contribution to the mean number of jobs in the system consisting of jobs in the Exponential class is precisely $\mathbf{E}\left[N^S_{Exp}\right] + \mathbf{E}\left[N^Q_{Exp}\right]$. The zero-sized jobs only contribute to the mean number in queue. However, since the scheduling policy is size-independent, the waiting time distribution of a zero-sized job is the same as the waiting time distribution of a job belonging to the Exponential class, but the arrival rate of zero-sized jobs is $\frac{C^2-1}{2}$ times the arrival rate of the Exponential class. Therefore, the contribution of the zero-sized jobs to the mean number in system is $\frac{C^2-1}{2}\mathbf{E}\left[N^Q_{Exp}\right]$, proving the proposition. □

In Section 2.4 we extend Proposition 3 to obtain an approximation for a GI/G/PS-MPL system involving the first two moments of the interarrival time and job size distributions.

2.3 The OPT-STATIC policy

We now introduce the OPT-STATIC policy to choose a near-optimal static MPL. The OPT-STATIC policy simply sets MPL = κ where κ denotes the MPL that minimizes the right hand side of (1):

$$\kappa = \arg\min_K \left\{ \mathbf{E}\left[N^S_{Exp}(K)\right] + \frac{C^2+1}{2}\mathbf{E}\left[N^Q_{Exp}(K)\right] \right\} \quad (2)$$

We now show that the OPT-STATIC policy is a good heuristic for minimizing the mean response time in an M/G/PS-MPL system with known mean arrival rate. In Figure 3, we present simulation results for the following three job size distributions *all with mean 1 and C^2=19*:

- Weibull distribution with scale parameter $\frac{1}{6}$ and shape parameter $\frac{1}{3}$.
- Bounded Pareto distribution with shape parameter $\alpha = 1.1$ and support $[0.182, 178.759]$.
- A two-phase hyperexponential (H_2) distribution whose parameters are chosen so that, r, the fraction of the total load constituted by the phase with the smaller mean, is 0.25.

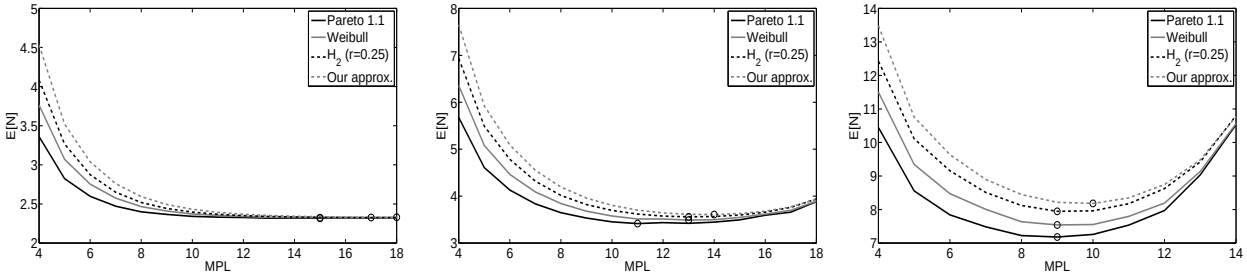
The results in Figure 3 assume that the state-dependent service rates of the PS server are given by the $\mu(n)$ curve shown in Figure 1. We will use the service rate curve shown in Figure 1 in all the numerical and simulation evaluations in this paper. In [11], we present detailed simulation results for more scenarios.

The main message of Figure 3 is that the optimal MPL can be much larger than the peak efficiency MPL of $K^* = 5$. For example, when $\lambda = 0.8$, the optimal static MPL for the bounded Pareto distribution is 11 with a resulting mean number of jobs around 3.4, while $K^* = 5$ results in 35% larger mean number of jobs at approximately 4.6. Second, as can be seen, even though approximation (1) is not extremely accurate at predicting the mean number of jobs in the system for general distributions (in fact, it is possible to show that no approximation based on only the first two moments can be), it is robust in predicting the optimal or near-optimal MPL. Our approximation recommends MPL = 14 and the mean number of jobs in the system using our recommended MPL is around 3.45.

Using approximation (1), it is easy to see why the mean number of jobs in the system is minimized at a larger MPL than the peak efficiency MPL of K^* when job sizes have high variability. To see this, start by considering the case of low variability: $C^2 = 1$. For this case, approximation (1) suggests that the optimal MPL is in fact K^*. As we increase the MPL beyond K^*, if the traffic intensity is not very high, $\mathbf{E}\left[N^Q_{Exp}\right]$ falls while $\mathbf{E}\left[N^S_{Exp}\right]$ increases. For a high enough C^2, the fall in $\frac{C^2+1}{2}\mathbf{E}\left[N^Q_{Exp}\right]$, and hence in the mean waiting time in the FCFS buffer, will be larger than the rise in $\mathbf{E}\left[N^S_{Exp}\right]$, which is the component representing the mean time to process a job at the PS server. Therefore, setting an MPL higher than K^*, and allowing small jobs to overtake the big jobs, leads to an overall reduction in the mean response time.

We would like to point out that the question of choosing the optimal multi-programming limit is closely related to the question of choosing the optimal number of servers in a multiserver system (that is, one fast vs. K slow servers), such as the $M/G/K$, but with a fundamentally different trade-off. In the presence of highly variable job sizes, one wants to choose a large number of servers in a multiserver system to prevent small jobs from getting blocked behind large jobs. Similarly, in the PS-MPL system, we want to choose a high MPL to allow small jobs to overtake large jobs. In both cases, we are limited in our ability to increase the parallelism due to capacity wastage. While in a multiserver system, capacity is wasted when there are less than K jobs in the system, in the PS-MPL system, capacity is wasted when the multi-programming limit K is set larger than the peak efficiency point K^*, and there are more than K^* jobs in the system. Therefore, in a multiserver system, high parallelism (large number of servers) is preferred when the traffic intensity is high, while in a PS-MPL system a high degree of parallelism (large MPL) is preferred when the traffic intensity is low.

2.4 Approximation for *GI/G/PS-MPL*



(a) $\lambda = 0.7$ (b) $\lambda = 0.8$ (c) $\lambda = 0.9$

Figure 3: The mean number of jobs in the system vs. MPL for the following distributions, all with mean 1 and SCV 19: (i) Bounded Pareto distribution with shape parameter 1.1 (ii) Weibull distribution (iii) Two-phase hyperexponential distribution with 25% of load constituted by the branch with the smaller mean. The arrival process considered is Poisson with the indicated mean arrival rate, λ. For reference, we have also shown our 2-moment approximation for the mean number of jobs in the system. The optimal MPL for each curve is shown with a circle.

PROPOSITION 5. *Let $\mathbf{E}[N]$ denote the mean number of jobs in a $GI/G/PS$-MPL system with state-dependent service rate $\mu(n)$ when there are n jobs at the PS server, MPL=K, a general job size distribution with mean 1 and SCV $C_s^2 \geq 1$, and a general interarrival time distribution with mean $\frac{1}{\lambda}$ and SCV $C_a^2 \geq 1$. Then,*

$$\mathbf{E}[N] \approx \mathbf{E}\Big[N'^S_{Exp}\Big] + \frac{C_s^2+1}{2}\mathbf{E}\Big[N'^Q_{Exp}\Big]$$

where $\mathbf{E}\Big[N'^S_{Exp}\Big]$ and $\mathbf{E}\Big[N'^Q_{Exp}\Big]$, denote, respectively, the mean number of jobs at the PS Server and in the FCFS Queue in a $BPP/M/PS$-MPL system with the same state-dependent service rates as the original $GI/G/PS$-MPL system, MPL=K, Exponential job size distribution with mean 1, mean arrival rate λ and i.i.d. geometric batch sizes with mean $\frac{C_s^2+C_a^2}{C_s^2+1}$. The expressions for $\mathbf{E}\Big[N'^S_{Exp}\Big]$ and $\mathbf{E}\Big[N'^Q_{Exp}\Big]$ are given by

$$\mathbf{E}\Big[N'^S_{Exp}\Big] = \frac{\sum_{i=1}^{K} i\cdot\phi_i + K\cdot\sum_{i=K+1}^{\infty}\phi_i}{1+\sum_{i=1}^{\infty}\phi_i} \quad (3)$$

$$\mathbf{E}\Big[N'^Q_{Exp}\Big] = \frac{\phi_{K+1}}{1+\sum_{i=1}^{\infty}\phi_i}\left(\frac{C_s^2+C_a^2}{(C_s^2+1)(1-\gamma)}\right)^2 \quad (4)$$

where

$$\phi_i = \begin{cases} \Pi_{j=1}^{i}\frac{\lambda\cdot(C_s^2+1)+\mu(j-1)\cdot(C_a^2-1)}{(C_s^2+C_a^2)\mu(j)} & 1\leq i\leq K \\ \phi_K\cdot\left(\frac{\gamma\cdot(C_s^2+1)+C_a^2-1}{C_s^2+C_a^2}\right)^{i-K} & i>K \end{cases}$$

and $\gamma = \frac{\lambda}{\mu(K)}$.

3. SELF-ADAPTIVE MPL CONTROL POLICIES

In the previous section, we considered the question of choosing the optimal static MPL under the assumption that the arrival process is Poisson, and that the mean arrival rate, λ, was known accurately. We begin this section by showing that the methodology of choosing a static MPL based on assuming a mean intensity for the Poisson arrival process is very fragile. In Table 1 we consider a Weibull job size distribution with mean 1 and $C^2 = 19$, and show the mean number of jobs in the system for various settings of MPL and the mean arrival rate λ. We assume the service rate curve shown in Figure 1 with $K^* = 5$. The optimal MPL in Table 1 varies from 15, when $\lambda = 0.65$, to 5, when $\lambda = 1.15$. In fact, choosing the optimal static MPL assuming a $\lambda \leq 0.85$ results in an unstable system when true $\lambda = 1.15$.

There can be two ways around this problem: The first approach is to robustly choose a single static MPL that works well for all λ. This necessarily implies operating the system at peak efficiency K^*, which we have already seen can be far from the optimal. The second approach is to learn the parameters of the arrival process and then choose the optimal static MPL for that particular arrival process. However, this approach will fail to adapt to variations in traffic on small time scales.

In this section, we are motivated by the question:

> *Are there light-weight, traffic-oblivious MPL control policies which perform as well as the traffic-aware optimal static MPL policies?*

By a traffic-oblivious control policy, we mean a policy that does not depend on knowing the arrival rate or the higher order characteristics of the arrival process.

In this section, we develop two dynamic MPL control policies - LIGHT-APPROX and POISSON-APPROX. Section 3.1 highlights the key ideas in our approach. Section 3.2 and Section 3.3, respectively, present the numerical algorithms involved in the construction of our traffic-oblivious dynamic MPL control policies LIGHT-APPROX and POISSON-APPROX. In Section 3.4 we evaluate our dynamic MPL control policies via simulations and demonstrate that our proposed MPL control policies exhibit robustness to both the traffic intensity and the burstiness of the arrival process.

3.1 Key Steps in Our Approach

Recall that, given a job size distribution, our goal is to obtain MPL control policies which are (i) light-weight: adjust the MPL based only on the instantaneous queue length, $Q(t)$, and the instantaneous MPL, $K(t)$, and (ii) traffic-oblivious: robust to variations in the arrival process.

To achieve our first goal, we consider a special class of job size distributions, the degenerate hyperexponential distribution (H^*), which is a mixture of an Exponential distribution, and a point mass at 0. Since the jobs of size 0 do not spend any time at the server, and due to the memoryless property of the Exponential distribution, $(Q(t), K(t))$ is

MPL	4	5	6	7	8	9	10	11	12	13	14	15	95% c.i.
$\lambda = 0.65$	2.960	2.471	2.258	2.149	2.092	2.058	2.040	2.029	2.023	2.020	2.018	**2.016**	$\pm$ 0.007
$\lambda = 0.75$	4.849	3.889	3.442	3.177	3.000	2.896	2.843	2.797	2.775	2.764	**2.755**	2.764	$\pm$ 0.020
$\lambda = 0.85$	8.493	6.797	6.025	5.542	5.226	4.984	4.903	**4.850**	4.929	5.019	5.217	5.589	$\pm$ 0.294
$\lambda = 0.95$	15.961	13.197	12.521	**12.162**	12.170	12.633	13.497	15.039	18.059	23.132	32.881	60.799	$\pm$ 2.483
$\lambda = 1.05$	33.929	**29.517**	31.082	34.989	40.705	52.554	72.724	126.900					$\pm$ 4.273
$\lambda = 1.15$	92.842	**87.189**	114.997	183.613									$\pm$ 5.606

Table 1: Numerical results for mean number of jobs in system for different values of MPL and arrival rates. The arrival process was Poisson, and the job size distribution was Weibull with mean 1, SCV 19. The optimal value for each setting of the mean arrival rate has been boldened.

a Markov process. This ensures that we can obtain a light-weight dynamic MPL control policy, since any optimal MPL control policy for the H^* job size distribution will only take decisions based on $(Q(t), K(t))$.

The next step in our approach is solving a stochastic dynamic programming problem to construct *families* of candidate dynamic MPL control policies. The LIGHT-APPROX and POISSON-APPROX policies differ in the family of candidate policies. Under LIGHT-APPROX, the family of candidate policies is a set, $\{\pi_p\}$, where a particular policy π_p is constructed by solving an optimal MPL control problem for an H^* job size distribution with parameter p (Eqn. (7)). Thus, while there is some unique $H^*(C^2)$ job size distribution that matches the first two moments of the true job size distribution (Definition 3), the family is constructed by looking at a range of H^* distributions. To solve the optimal control problem, we assume that we start in some initial state (Q_0, K_0), and find the policy that minimizes the sum of response time of jobs in the system given that there are no further arrivals. In the case of POISSON-APPROX, the family of candidate policies is the set, $\{\pi_{\lambda_p}\}$, where a particular policy π_{λ_p} is obtained by solving an optimal control problem for a Poisson arrival process with intensity λ_p and the $H^*(C^2)$ job size distribution to minimize the time-average mean number of jobs in the system.

The final step in our approach is choosing one member from the family of candidate dynamic policies, so that the chosen policy is robust to the arrival process. To achieve this goal, we evaluate the candidate policies in the family for a Poisson arrival process with rate $\lambda \in [\underline{\lambda}, \overline{\lambda}]$ and $H^*(C^2)$ job size distribution. Let $\mathbf{E}[N^*(\lambda)]$ denote the mean number of jobs in the system for Poisson arrival process with intensity λ, and $H^*(C^2)$ job size distribution, under the OPT-STATIC policy. The quantity $\mathbf{E}[N^*(\lambda)]$ is given by Proposition 4. Let $\mathbf{E}[N^\pi(\lambda)]$ denote the mean number of jobs in the system for the $H^*(C^2)$ job size distribution and Poisson arrival process with intensity λ under a dynamic MPL control policy π. We define the worst-case relative error for a policy π as:

$$\epsilon(\pi) = \max_{\lambda \in [\underline{\lambda}, \overline{\lambda}]} \frac{\mathbf{E}[N^\pi(\lambda)] - \mathbf{E}[N^*(\lambda)]}{\mathbf{E}[N^*(\lambda)]} \tag{5}$$

Given a family of candidate policies $\{\pi_a\}$ with parameter a taking values in some set A, we choose the policy that minimizes the worst case relative error:

$$a^* = \arg\min_{a \in A} \epsilon(\pi_a) \tag{6}$$

Thus, in our case, π_{p^*} denotes the LIGHT-APPROX policy, and $\pi_{\lambda_p^*}$ denotes the POISSON-APPROX policy.

3.2 The LIGHT-APPROX policy

As a first step towards deriving the LIGHT-APPROX policy, we begin in Section 3.2.1 by formulating and solving a light-traffic optimal MPL control problem. We find that the solution to this problem exhibits both a fluid component, to guarantee stability, and a stochastic component, to handle variability in job sizes. In Section 3.2.2, we use the solution of the light-traffic optimal control problem to construct a family, $\{\pi_p\}$, of simple, light-weight MPL control policies, and in Section 3.2.3 we sketch the use of Matrix-Geometric methods to evaluate this family of candidate policies to enable selection of the appropriate policy, LIGHT-APPROX.

3.2.1 A light-traffic optimal control problem

In this section we solve an optimal light-traffic MPL control problem parameterized by p, by considering the following degenerate hyperexponential job size distribution :

$$H^*(p) \sim \begin{cases} 0 & \text{with probability } p \\ \text{Exp}\,(1) & \text{with probability } 1-p \end{cases} \tag{7}$$

We assume that we start our PS-MPL system in some state (Q_0, K_0) at time $t = 0$, where a departure has taken place at time $t = 0^-$. The state variable Q_0 denotes the queue length at $t = 0^-$ and K_0 is one more than the number of jobs at the PS server left behind by the last departure. We assume that multiple zero-sized jobs admitted at the same time leave together. Thus K_0 does not necessarily denote the MPL at time $t = 0^-$. However, by our assumption of an $H^*(p)$ job size distribution, each of the $(K_0 - 1)$ jobs at the server has remaining service requirement independent and identically distributed as Exp(1). Note that while the zero-sized jobs do not spend any time at the server, they still experience delays while waiting in the FCFS buffer. We *assume that there are no more arrivals* (hence the light-traffic). We can now take one of the following actions at time $t = 0$:

1. **Decrease MPL**: We do not admit another job from the queue into the PS server, decreasing the MPL to $K_0 - 1$.

2. **Keep MPL same**: We admit only one job from the queue into the PS server to replace the departing job, maintaining the MPL at K_0.

3. **Increase MPL by** k: We admit $k + 1$ jobs from the queue into the PS server, increasing the MPL to $K_0 + k$.

Our aim is to take the optimal action in each state so as to achieve the following **goal**:

Minimize the expected sum of response times of jobs present in the system at time $t = 0$, given that there are no further arrivals.

If our goal was to minimize the time until the system empties, the optimal control would be to operate at MPL of K^*. However our performance metric is the mean response time. Note that we do not allow preempting an executing job to decrease the MPL. This is important because in a transaction processing system, for instance, killing an executing task involves unrolling the execution trace for the task and is significantly expensive. In our framework, we can only alter the MPL when a job departs, and hence we assume that there are no costs associated with changing the MPL.

The solution of the above optimal-control problem can be obtained in a straightforward fashion via stochastic dynamic programming. To do so, we associate a cost function $c(Q, K)$ with each state (Q, K), which represents the optimal expected sum of response times, given that we start in state (Q, K) at time $t = 0$, and an action function $\pi(Q, K)$, representing the optimal action in state (Q, K). The function $\pi(Q, K)$ takes values in the range $\{-1, 0, 1, 2, \ldots\}$ with -1 representing the action 'decrease MPL', 0 representing the action 'keep MPL same' and $k > 0$ representing the action 'increase MPL by k'.

The cost of the states with zero queue length is simply:

$$c(0, K) = \sum_{i=1}^{K-1} \frac{i}{\mu(i)} \tag{8}$$

To see why the above is true, note that since the queue is empty and we do now allow preemption of executing jobs, the cost of state $(0, K)$ is the expected sum of response times of the $K - 1$ jobs executing at the server. The mean time until the departure of the first job is given by $\frac{1}{\mu(K-1)}$ since the server is processing at rate $\mu(K - 1)$. The time until the first departure gets added to the response time of all the jobs in the system, and contributes $\frac{K-1}{\mu(K-1)}$ to $c(0, K)$, and so on for subsequent departures.

We represent by $c_{-1}(Q, K)$ the cost of state (Q, K) given that we take action 'decrease MPL' in state (Q, K). Similarly, $c_k(Q, K)$ ($k \in \{0, \ldots, Q-1\}$) denotes the cost of state (Q, K) given that we take action 'increase MPL by k' in state (Q, K). Given $c_{-1}(Q, K)$ and $c_k(Q, K)$, the optimal action $\pi(Q, K)$ and the cost function $c(Q, K)$ are:

$$\pi(Q, K) = \arg\min_{\delta} c_\delta(Q, K) \quad \delta \in \{-1, \ldots, Q-1\} \tag{9}$$

$$c(Q, K) = c_{\pi(Q,K)}(Q, K) \tag{10}$$

The function $c_{-1}(Q, K)$ is given by:

$$c_{-1}(Q, K) = \frac{Q + K - 1}{\mu(K - 1)} + c(Q, K - 1) \tag{11}$$

and $c_k(Q, K)$ is given by:

$$c_k(Q, K) = \left[\frac{Q + K - 1}{\mu(K + k)} + c(Q - k - 1, K + k)\right] \cdot (1 - p)^{k+1} + \sum_{i=1}^{k+1} c(Q - k - 1, K + k + 1 - i) \cdot \binom{k+1}{i} (1 - p)^{k+1-i} p^i \tag{12}$$

In deriving the last equation, we have made use of the assumption that if multiple zero-sized jobs are admitted simultaneously, then they all leave together. This maintains the invariant that the K in state descriptor (Q, K) is one larger than the number of jobs at the server belonging to the Exponential class, and we do not have to keep track or estimate the number of zero-sized jobs.

While in the problem formulation above, we have not imposed an upper bound on k, in practice we restrict $k \leq \Delta_{max}$ to prevent sudden jumps in MPL. For all the simulation results in this paper, we set $\Delta_{max} = 1$.

3.2.2 A family of traffic-oblivious MPL control policies

In Section 3.2.1 we formulated an optimal control problem parameterized by p, the fraction of zero-sized jobs in the $H^*(p)$ job size distribution. By varying the parameter p, we obtain a family of MPL control policies. Let π_p denote the action function for the control problem with parameter p. Figure 4 shows the structure of π_p for $p = 0.3$ and $p = 0.5$ and the service rate curve shown in Figure 1. For example, if the current state is $(Q = 21, K = 10)$, under the $p = 0.3$ policy, the control is to decrease the MPL to 9 by not admitting a new job, while under $p = 0.5$ policy, the optimal control is to increase the MPL to 11 by admitting two jobs. The structure of the optimal solution has some interesting features:

1. For a given p, there is some minimum queue length $Q(p)$ such that the optimal action for $Q > Q(p)$ is to operate at the peak efficiency point. In Figure 4(a), $Q(p) = 20$ and the optimal control for $Q > Q(p)$ is to attain the peak efficiency MPL of $K^* = 5$. We call this the *fluid component* of the control policy. This fluid component provides robustness to the dynamic MPL policy against high arrival rates. Further, as p increases, the threshold $Q(p)$ increases.

2. As the queue length decreases, the *stochastic component* of the control takes over, gradually increasing the MPL to a point with lower service rate than the most efficient point. This stochastic component gives our MPL control policy the ability to combat the job-size-variability when the traffic intensity is low.

The structure of the optimal control is quite intuitive. Whenever a decision to increase the MPL has to be taken, there are two scenarios: (i) with probability p the admitted job is of size zero in which case the decrease in server speed does not hurt any one, and (ii) with probability $1 - p$, the admitted job belongs to the Exponential class and in this case adds to the waiting time of everyone in the queue. If we define the 'threshold queue length' to be the point when we should increase the MPL and move to a less efficient service rate, then we see that this threshold queue length is an increasing function of p.

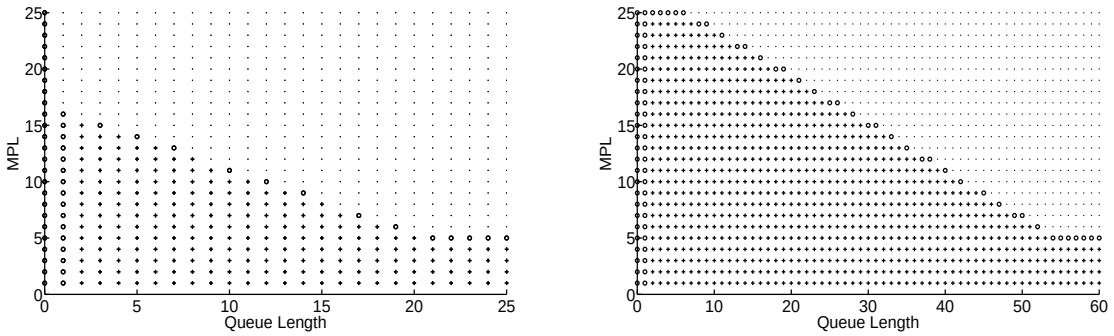
Given any action function π, we can translate it into a dynamic MPL control policy via the procedure in Figure 5.

The LIGHT-APPROX control policy for a distribution with SCV C^2 is now chosen to be π_{p^*} such that:

$$p^* = \arg\min_{p} \epsilon(\pi_p) \tag{13}$$

where $\epsilon(\cdot)$ is given by (5). Experimentally, it suffices to carry out the optimization over a small set of parameters p (at a coarse granularity).

3.2.3 Evaluation of dynamic MPL control policies via Matrix-geometric analysis



(a) $p = 0.3$ (b) $p = 0.5$

Figure 4: The structure of the Light-Approx control policy for two values of the parameter p and $\Delta_{max} = 1$. A '+' indicates 'increase MPL', and a 'o' indicates 'keep MPL same'. At every other point, the optimal control is to decrease MPL.

Algorithm MPL_control(π)
Case: New arrival

- Let Q be the queue length and K be the MPL immediately after the arrival.
- Let $\pi(Q, K+1) = k$
  - if $k \geq 0$: admit $k+1$ jobs from the head of the FCFS buffer into the server and increase MPL to $K + k + 1$
  - if $k < 0$: do nothing

Case: Departure

- Let Q be the queue length and K be the MPL immediately before the departure.
- Let $\pi(Q, K) = k$
  - if $k \geq 0$: admit $k+1$ jobs from the head of the FCFS buffer into the server and set MPL to $K + k$
  - if $k < 0$: reduce MPL to $K - 1$ by not admitting any job from the FCFS buffer

Figure 5: The dynamic MPL control policy obtained from the action function π.

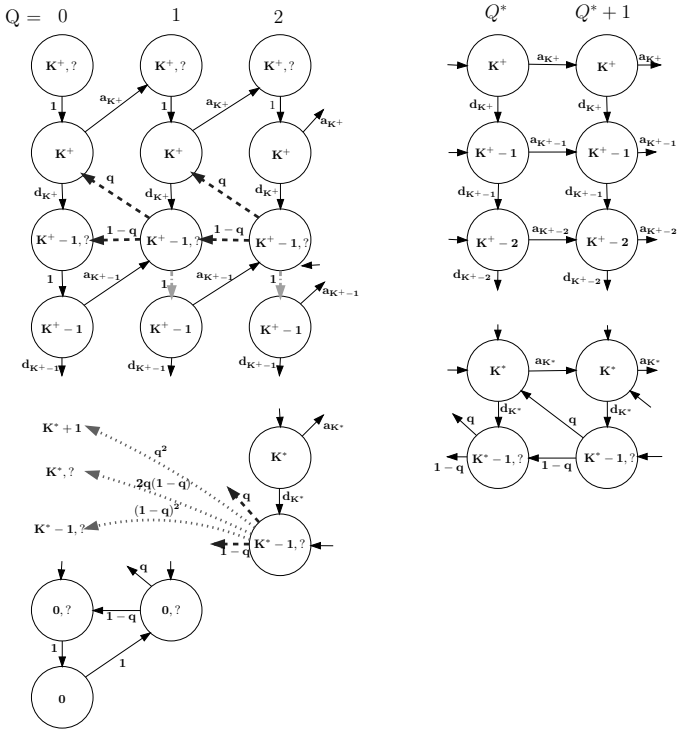


Figure 6: The embedded Markov chain for evaluation of dynamic MPL control policies. We use a_n to denote $\frac{\lambda}{\lambda + q \cdot \mu(n)}$ and $d_n = 1 - a_n$. For decision states with multiple alternatives (e.g., $(1, K^+ - 1, ?)$ and $(2, K^* - 1, ?)$), the dash-dotted arcs correspond to the decision to not admit any jobs, dashed arcs correspond to the decision to admit one job, and dotted arcs correspond to the decision to admit two jobs.

In this section, we outline a method to numerically evaluate the mean number of jobs, $\mathbf{E}[N^\pi(\lambda)]$, for a dynamic MPL control policy π under the assumption of the $H^*(C^2)$ job size distribution (Definition 3) and a Poisson arrival process of intensity λ. Note that in Proposition 4 with static MPL, we were able to simplify the analysis of the $H^*(C^2)$ job size distribution by ignoring the zero-sized jobs and focusing on the exponential class. This was because the admission control policy was independent of the queue-length. However, with a dynamic policy that looks at the queue-length, we need to keep track of how many zero-sized jobs are in the system. For succinctness, let $q = \frac{2}{C^2+1}$.

Assuming that under the dynamic policy π, there is some queue-length Q^* such that the optimal control for any queue length $Q \geq Q^*$ is to operate at the highest efficiency point K^*, we can express the system as a Markov chain with a repeating structure. The states of the Markov chain are pairs (Q, K) with Q denoting the queue length, and K denoting the number of jobs of the exponential class at the server. However, due to the zero-sized jobs, we can have arbitrarily big drops in Q. For example, if we are in state $(Q = 10, K = 5)$ and a departure takes place, and if all the jobs in the queue have size 0, which happens with non-zero probability, we jump to state $(Q = 0, K = 4)$. To

take care of this problem, we introduce *decision* states represented as $(Q, K, ?)$. We transition to the decision state $(Q, K, ?)$ immediately after a departure takes place from the state $(Q, K+1)$, or if an arrival takes place while in state $(Q-1, K)$ and $Q < Q^*$. The state $(Q, K, ?)$ implements the admission control policy π, as well as handling zero-sized jobs, because now the jumps are bounded. For example, if the control in state $(Q, K, ?)$ is to admit 1 job, then with probability $(1-q)$ the job is of size 0, and we transition to $(Q-1, K, ?)$; otherwise, with probability q we transition to $(Q-1, K+1)$. However, the rate of transitioning from the decision states is infinite. Thus we will find it suitable instead to work in the framework of Semi-Markov processes. We will consider the embedded discrete time Markov chain where the transitions correspond to arrivals, departure and decisions taken in decision states in the original continuous time system. The embedded Markov chain is shown in Figure 6. We then solve for the stationary distribution of this embedded Markov chain via Matrix-Geometric method. We would like to point out that due to the special structure of the Markov chain in Figure 6 (the backward transition matrix is of rank 1), the rate matrix involved in the Matrix-geometric solution has an explicit solution in our case [20]. Finally, we obtain the stationary distribution of the number of jobs in the system by multiplying the probability of being in a state in the embedded chain with the mean residence time in that state, and normalizing.

3.3 The POISSON-APPROX policy

The POISSON-APPROX policy is defined by constructing a family $\{\pi_{\lambda_p}\}$, where the candidate policy π_{λ_p} is obtained as follows: We consider a Poisson arrival process of intensity λ_p and the $H^*(C^2)$ job size distribution, and solve the optimal dynamic MPL control problem to minimize the mean number of jobs. The policy π_{λ_p} is computed via the method of policy iteration, explained in Appendix A. Figure 7 shows the structure of π_{λ_p} for $\lambda_p = 0.95$ and $\lambda_p = 1.05$. The POISSON-APPROX MPL control policy is now chosen to be $\pi_{\lambda_p^*}$ where:

$$\lambda_p^* = \arg\min_{\lambda_p} \epsilon(\pi_{\lambda_p}) \tag{14}$$

where $\epsilon(\cdot)$ is defined in (5). As in the case of LIGHT-APPROX, it suffices to carry out the above optimization at a coarse granularity.

3.4 Performance Evaluation

In this section we show via simulations that our dynamic MPL control policies proposed in Sections 3.2 and 3.3 guarantee robustness against both misestimation of traffic intensity, and against higher order characteristics of the arrival process, such as the burstiness.

3.4.1 Robustness against traffic intensity estimation

We will now evaluate the LIGHT-APPROX and POISSON-APPROX policies for a Poisson arrival process with unknown mean arrival rate, λ, and compare them against the OPT-STATIC policy that is given the exact mean arrival rate. To do this, we show the mean number of jobs, $\mathbf{E}[N]$, under different arrival rates, obtained via simulations. Recall that Table 1 shows these results for the Weibull job size distribution and various values of static MPLs. In Table 2 we show the results for the mean number of jobs for the same Weibull job size distribution under the LIGHT-APPROX policy, as a function of λ and the parameter p of the family $\{\pi_p\}$ of candidate policies. The optimization procedure (13) sets $p^* = 0.25$ from among the values shown in the table (column highlighted). Observe that the LIGHT-APPROX policy gives near optimal performance for each arrival rate as compared to Table 1 for λ up to 1.05 with approximately 13% larger mean number of jobs in the system than the optimal traffic-aware static policy when $\lambda = 0.85$. On the other hand, a single robustly chosen static MPL necessarily has to operate at the peak efficiency point and, as Table 1 shows, exhibits 41% larger mean response time than the optimal traffic-aware static policy when $\lambda = 0.75$.

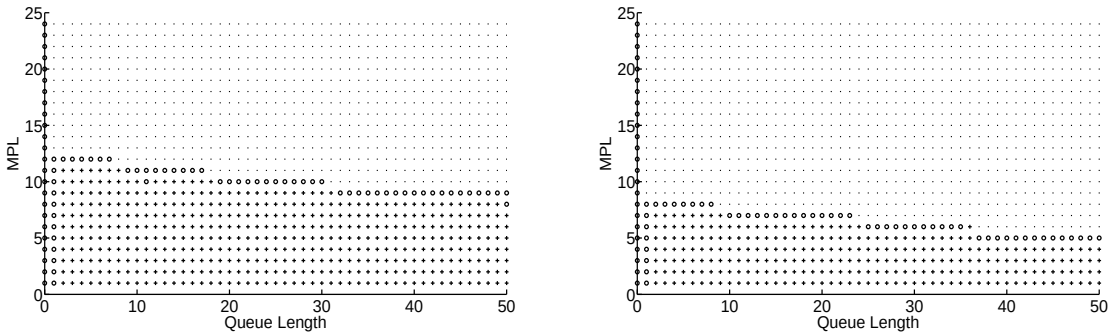
Table 3 shows simulation results for the mean number of jobs with the POISSON-APPROX MPL control policy for various values of the parameter λ_p for the family $\{\pi_{\lambda_p}\}$ of candidate policies. The optimization procedure (14) sets $\lambda_p^* = 0.95$ from among the values shown in the table (column highlighted). The POISSON-APPROX policy also achieves near-optimal performance for each arrival rate as compared to Table 1 with approximately 19.5% larger mean number of jobs in the system than the optimal traffic-aware static policy when $\lambda = 1.15$. Note that for these results, we have not completely optimized the λ_p parameter, and the performance of the POISSON-APPROX policy is likely to improve further.

While we have seen that both dynamic policies are far superior than any static policy when the mean arrival rate is not known, looking both at Tables 2 and 3, one can observe that neither dynamic policy significantly outperforms the OPT-STATIC policy if the mean arrival rate is known.

3.4.2 Robustness against burstiness in arrival process with unknown arrival rate

We now evaluate the robustness of our MPL control policies against burstiness of the arrival process when the mean arrival rate is not known. To do so, we choose a batch Poisson arrival process (BPP). The batch sizes were i.i.d. geometric with mean 5. Table 4 shows the results for the mean number of jobs in the system with Weibull job size distribution for various settings of static MPL and mean arrival rate λ of the arrival process. From Table 4, we see that when the arrival rate is not known, a robustly chosen static policy has to operate at $K^* = 5$, which results in 50% higher mean number of jobs than the optimal traffic-aware static policy when the mean arrival rate is $\lambda = 0.65$. Therefore a bursty arrival process can exacerbate the inadequacy of static MPL policies when the mean arrival rate is not known.

Table 5 shows the results for mean number of jobs in the system for the same setting as Table 4 for the LIGHT-APPROX MPL control policy as a function of the parameter p of the family $\{\pi_p\}$ of candidate policies for various values of the mean arrival rate λ. The column for the parameter chosen by the LIGHT-APPROX policy has been highlighted. From Table 5, we find that LIGHT-APPROX policy is also robust to burstiness, while yielding at worst 25% higher mean response time than the optimal traffic-aware static MPL policy. Therefore, the LIGHT-APPROX policy is simultaneously robust to both the mean arrival rate and burstiness of the arrival process. The LIGHT-APPROX policy with parameter $p = 0.3$ outperforms the policy with $p = 0.25$ for the chosen setting, but as noted earlier, this is due to the fact that we have not optimized the parameter completely.



(a) $\lambda_p = 0.95$ (b) $\lambda_p = 1.05$

Figure 7: The structure of the Poisson-Approx control policy for two values of the parameter λ_p and $\Delta_{max} = 1$. A '+' indicates 'increase MPL', and a 'o' indicates 'keep MPL same'. At every other point, the optimal control is to decrease MPL.

p	0.2	**0.25**	0.3	0.35	0.4	0.45	0.5	95% conf. int.
$\lambda = 0.65$	2.146	**2.068**	2.037	2.026	2.020	2.018	2.017	± 0.004
$\lambda = 0.75$	3.268	**3.034**	2.898	2.842	2.803	2.806	2.804	± 0.015
$\lambda = 0.85$	5.939	**5.502**	5.222	5.138	5.188	5.330	5.497	± 0.049
$\lambda = 0.95$	12.473	**12.315**	12.312	12.838	13.830	15.480	17.035	± 0.201
$\lambda = 1.05$	29.807	**30.731**	32.445	35.981	40.748	46.836	53.344	± 0.381
$\lambda = 1.15$	89.075	**93.378**	99.875	108.300	120.120	132.354	143.678	± 1.724

Table 2: Simulation results for mean number of jobs, $\mathbf{E}[N]$, for different parameters p of the Light-Approx policy and arrival rates, λ. The arrival process is Poisson(λ), and the job size distribution is Weibull with mean 1, SCV 19.

λ_p	0.65	0.75	0.85	**0.95**	1.05	1.15	95% conf. int.
$\lambda = 0.65$	2.019	2.017	2.021	**2.032**	2.134	2.474	± 0.005
$\lambda = 0.75$	2.792	2.768	2.778	**2.843**	3.192	3.890	± 0.015
$\lambda = 0.85$	5.821	5.241	4.890	**4.981**	5.656	6.776	± 0.068
$\lambda = 0.95$	20.992	16.737	13.269	**11.874**	12.100	13.210	± 0.183
$\lambda = 1.05$	66.047	53.765	40.952	**33.487**	29.675	29.464	± 0.536
$\lambda = 1.15$	166.720	149.052	124.863	**104.401**	91.015	86.473	± 1.967

Table 3: Simulation results for mean number of jobs, $\mathbf{E}[N]$, for different parameters λ_p of the Poisson-Approx policy and arrival rates λ. The arrival process is Poisson(λ), and the job size distribution is Weibull with mean 1, SCV 19.

Table 6 shows the results for mean number of jobs in the system under the same setting for the Poisson-Approx MPL control policy as a function of the parameter λ_p of the family $\{\pi_{\lambda_p}\}$ of candidate policies. The column for the parameter chosen by the Poisson-Approx policy has been highlighted. From Table 6, we find that the Poisson-Approx policy yields at worst 13% higher mean response time than the optimal traffic aware static MPL policy. Thus while both our policies are robust to bursty arrival processes, the Poisson-Approx policy seems to marginally outperform the Light-Approx policy. These observations also hold true for the simulation experiments presented in [11].

While we have demonstrated that our dynamic policies are much more robust than any static policy in handling burstiness when the mean arrival rate is not known, comparing Tables 1 and 4, we see that, surprisingly, if the mean arrival rate of the arrival process is known, the Opt-Static policy which optimizes for a Poisson arrival process with the given mean arrival rate remains near-optimal for a bursty arrival process.

4. CONCLUSION

In this paper we consider the problem of admission control in a resource-sharing system with load-dependent service rates, when the job size distribution is highly variable. We demonstrate, that counter to common wisdom, imposing a static multi-programming limit (MPL) at the peak efficiency point is not always optimal for minimizing the mean response time, and propose a simple rule to choose the optimal static MPL under the assumption that traffic is Poisson with a known arrival rate.

Next, we show that a static MPL policy can not be robust to varying traffic patterns, such as the variability in the mean arrival rate. We propose two simple MPL control policies, Light-Approx and Poisson-Approx, that adjust the MPL based on knowledge of only the instantaneous queue length, not the arrival process. We show that our dynamic MPL control policies exhibit robustness to both an unknown mean arrival rate and to burstiness in the arrival process. Specifically, our dynamic control policies result in mean response times within 25% of the mean response time of the optimal traffic-aware static MPL policy.

Our analysis illuminates several key ideas that are important in finding the optimal MPL. While a high arrival rate requires setting the MPL at the point of peak efficiency, when the arrival rate is moderate, we find that the optimal MPL is typically much higher than the peak efficiency point. The reason is that allowing a higher degree of parallelism (higher MPL) can alleviate the effect of high job size variability. In situations where the mean arrival rate is

MPL	4	5	6	7	8	9	10	11	12	13	14	15	95% c.i.
$\lambda = 0.65$	5.802	4.853	4.350	3.995	3.755	3.586	3.464	3.374	3.316	3.269	3.250	**3.232**	$\pm$ 0.015
$\lambda = 0.75$	9.287	7.799	7.093	6.531	6.176	5.871	5.691	5.564	5.490	**5.471**	5.472	5.583	$\pm$ 0.083
$\lambda = 0.85$	15.438	13.205	12.295	11.677	11.383	**11.200**	11.225	11.437	11.868	12.758	13.908	15.854	$\pm$ 0.905
$\lambda = 0.95$	27.245	24.048	**23.860**	24.363	25.173	26.846	29.453	34.285	41.082	54.117	78.139	141.179	$\pm$ 3.495
$\lambda = 1.05$	53.044	**49.187**	53.368	60.676	71.380	90.525	130.344	210.054					$\pm$ 4.932
$\lambda = 1.15$	136.640	**131.356**	176.238	274.395									$\pm$ 7.308

Table 4: Simulation results for mean number of jobs, $\mathbf{E}[N]$, for different values of MPL and mean arrival rates λ. The arrival process is a batch Poisson process where the arriving batch sizes are geometrically distributed with mean 5, and the job size distribution is Weibull with mean 1 and SCV 19. The optimal value for each setting of the mean arrival rate is boldened.

p	0.2	**0.25**	0.3	0.35	0.4	0.45	0.5	95% conf. int.
$\lambda = 0.65$	4.396	**4.039**	3.752	3.537	3.414	3.349	3.327	$\pm$ 0.017
$\lambda = 0.75$	7.301	**6.806**	6.395	6.100	5.919	5.901	6.005	$\pm$ 0.040
$\lambda = 0.85$	12.667	**12.262**	11.880	11.776	12.024	12.670	13.457	$\pm$ 0.104
$\lambda = 0.95$	23.836	**23.714**	24.053	24.974	26.619	29.422	32.592	$\pm$ 0.312
$\lambda = 1.05$	49.542	**50.419**	52.110	55.588	60.669	67.073	74.570	$\pm$ 0.569
$\lambda = 1.15$	133.778	**137.341**	141.705	149.050	158.226	171.736	182.818	$\pm$ 2.458

Table 5: Simulation results for mean number of jobs, $\mathbf{E}[N]$, for different parameters p of the Light-Approx policy and mean arrival rates λ. The arrival process is a batch Poisson process where the arriving batch sizes are geometrically distributed with mean 5, and the job size distribution is Weibull with mean 1 and SCV 19.

λ_p	0.65	0.75	0.85	**0.95**	1.05	1.15	95% conf. int.
$\lambda = 0.65$	3.271	3.264	3.309	**3.480**	4.073	4.851	$\pm$ 0.017
$\lambda = 0.75$	5.929	5.641	5.636	**5.849**	6.800	7.828	$\pm$ 0.035
$\lambda = 0.85$	14.033	12.427	11.426	**11.177**	12.170	13.170	$\pm$ 0.125
$\lambda = 0.95$	36.054	30.402	26.075	**23.898**	23.299	23.932	$\pm$ 0.244
$\lambda = 1.05$	84.019	71.672	60.758	**54.104**	49.372	48.831	$\pm$ 0.510
$\lambda = 1.15$	199.389	180.815	162.219	**148.864**	135.175	131.729	$\pm$ 2.239

Table 6: Simulation results for mean number of jobs, $\mathbf{E}[N]$, for different parameters λ_p of the Light-Approx policy and arrival rates λ. The arrival process is a batch Poisson process where the arriving batch sizes are geometrically distributed with mean 5, and the job size distribution is Weibull with mean 1 and SCV 19.

not known, dynamic MPL policies are needed to moderate between the effect of the arrival rate and the job size variability: when the instantaneous arrival rate is high, the effect of mean arrival rate dominates the effect of C^2, whereas when the arrival rate is low, the effect of C^2 is more dominant.

The techniques presented herein for obtaining simple and robust optimal control policies are applicable to more general stochastic settings. While the majority of literature on robust dynamic control focuses on solving the corresponding optimal control problem in the fluid regime, our techniques allow one to obtain optimal control policies which exhibit components of both stochastic and fluid control.

5. REFERENCES

[1] M. Abouzour. Automatically tuning database server multiprogramming level. Master's thesis, University of Waterloo, 2007.

[2] R. Agrawal, M. J. Carey, and M. Livny. Models for studying concurrency control performance: alternatives and implications. *SIGMOD Rec.*, 14(4):108–121, 1985.

[3] B. Avi-Itzhak and S. Halfin. Expected response times in a non-symmetric time sharing queue with a limited number of service positions. *In Proceedings of ITC*, 12:5.4B.2.1–7, 1988.

[4] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1-2. Athena Scientific, 3rd edition, 2007.

[5] R. Blake. Optimal control of thrashing. In *Proceedings of the 1982 ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems*, 1982.

[6] S. Borst and R. Núñez-Queija. Introduction to special issue on queueing models for fair resource sharing. *Queueing Syst.*, 53(1-2):5–6, 2006.

[7] E. G. Coffman, Jr., R. R. Muntz, and H. Trotter. Waiting time distributions for processor-sharing systems. *J. Assoc. Comput. Mach.*, 17:123–130, 1970.

[8] P. J. Denning, K. C. Kahn, J. Leroudier, D. Potier, and R. Suri. Optimal multiprogramming. *Acta Informatrica*, 7:197–216, 1976.

[9] S. Elnikety, E. Nahum, J. Tracy, and W. Zwaenepoel. A method for trasparent admission control and request schedulin in e-commerce web sites. In *World-Wide-Web Conference*, 2004.

[10] A. Fredericks. Approximations for customer viewed delays in multiprogrammed, transaction oriented computer systems. *Bell System Technical Journal*, 59(9):1559–1574, 1980.

[11] V. Gupta and M. Harchol-Balter. Self-adaptive admission control policies for resource-sharing systems. Technical Report CMU-CS-09-115, School of Computer Science, Carnegie Mellon University, 2009.

[12] H.-U. Heiss and R. Wagner. Adaptive load control in transaction processing systems. In *Proceedings of the 17th International Conference on Large Data Bases (VLDB)*, 1991.

[13] A. Kamra, V. Misra, and E. M. Nahum. Yaksha: A self-tuning controller for managing the performance of

3-tiered web sites. In *Twelfth IEEE International Workshop on Quality of Service (IWQOS)*, 2004.

[14] L. Kleinrock. Analysis of a time-shared processor. *Naval research logistics quarterly*, pages 59–73, 1964.

[15] L. Kleinrock. Time-shared systems: A theoretical treatment. *J. Assoc. Comput. Mach.*, 14:242–261, 1967.

[16] L. Kleinrock. *Queueing Systems; Volume 2: Computer Applications*. Wiley, New York, 1976.

[17] A. Lee and P. Longton. Queueing process associated with airline passenger check-in. *Operations Research Quarterly*, 10:56–71, 1959.

[18] M. Neuts. *Matrix-geometric solutions – An algorithmic approach*. The Johns Hopkins University Press, Baltimore, MD, 1981.

[19] M. Nuyens and W. van der Weij. Monotonicity in the limited processor sharing queue. Technical Report PNA-E0802, CWI, 2008.

[20] V. Ramaswami and G. Latouche. A general class of Markov processes with explicit matrix-geometric solutions. *OR Spektrum*, 8:209–218, 1986.

[21] K. Rege and M. Sengupta. Sojourn time distribution in a multiprogrammed computer system. *AT&T Tech. J.*, 64:1077–1090, 1985.

[22] R. Righter, J. G. Shanthikumar, and G. Yamazaki. On extremal service disciplines in single-stage queueing systems. *J. Appl. Probab.*, 27(2):409–416, 1990.

[23] D. M. Ritchie and K. Thompson. The Unix time-sharing system. *C. ACM*, 17(7):365–375, 1974.

[24] B. Schroeder, M. Harchol-Balter, A. Iyengar, E. Nahum, and A. Wierman. How to determine a good multi-programming level for external scheduling. In *Proceedings of the 22nd International Conference on Data Engineering*, Atlanta, GA, April 2006.

[25] W. van der Weij, S. Bhulai, and R. van der Mei. Optimal scheduling policies for the limited processor sharing queue. Technical Report WS2008-5, Department of Mathematics, Vrije University, 2008.

[26] M. Welsh, D. Culler, and E. Brewer. Seda: an architecture for well-conditioned, scalable internet services. *SIGOPS Oper. Syst. Rev.*, 35(5):230–243, 2001.

[27] S. F. Yashkov. Processor-sharing queues: some progress in analysis. *Queueing Systems Theory Appl.*, 2(1):1–17, 1987.

[28] J. Zhang and B. Zwart. Steady state approximations of limited processor sharing queues in heavy traffic. *Submitted for publication*.

[29] A. P. Zwart and O. J. Boxma. Sojourn time asymptotics in the $M/G/1$ processor sharing queue. *Queueing Systems Theory Appl.*, 35(1-4):141–166, 2000.

APPENDIX

A. POLICY ITERATION TO CONSTRUCT CANDIDATE POISSON-APPROX POLICIES

The goal of this section is to explain the policy iteration algorithm to find the optimal MPL control policy π_{λ_p}, for a Poisson arrival process with intensity λ_p and the $H^*(C^2)$ job size distribution matching the true job size distribution (Definition 3).

Let us first recall how policy iteration works [4]. We begin with some MPL control policy π^0 (in our case, a good initial policy is the threshold MPL policy which operates at the peak efficiency point K^*). Let γ^0 be the average cost (in our case the mean number of jobs in the system) of this policy. We then define the differential cost function $h^0(\cdot)$ associated with each state, where $h^0(s_i)$ denotes the differential cost to reach some state s_0 starting in state s_i under π^0. That is, $h^0(s_i)$ denotes the difference between the mean total cost to reach state s_0, and the product of γ^0 and the mean total time to reach state s_0, given that we start in state s_i. The vector of differential costs, $h^0(\cdot)$, and the average cost, γ^0, are obtained by solving the following linear system of equations:

$$h^0(s_0) = 0$$
$$h^0(s_i) = c(s_i)\tau(s_i) - \gamma^0\tau(s_i) + \sum_j p_{ij}(\pi^0(s_i))h(s_j)$$

where $\tau(s_i)$ is the mean residence time in state s_i, $c(s_i)$ is the cost per unit of time in state s_i, and $p_{ij}(\pi^0(s_i))$ represents the probability that we transition from state s_i to s_j when control $\pi^0(s_i)$ is applied in state s_i. This is called the policy evaluation step. We then perform the policy improvement step to obtain the policy π^1. To do this, for each state s_i, we choose $\pi^1(s_i)$ as the control which satisfies:

$$c(s_i)\tau(s_i) - \gamma^0\tau(s_i) + \sum_j p_{ij}(\pi^1(s_i))h(s_j)$$
$$= \min_{a\in A_i}\left[c(s_i)\tau(s_i) - \gamma^0\tau(s_i) + \sum_j p_{ij}(a)h(s_j)\right]$$

where A_i is the set of possible actions in state i. We then keep performing policy evaluation and improvement until two consecutive policies are the same, or have the same average cost.

The policy iteration step can be easily performed once the policy evaluation step is performed. The policy evaluation step is clearly tractable when the state space is finite. We now show it is also tractable when the state space is infinite but repeating, obeying the conditions for Matrix-Geometric analysis. In the remaining section, we focus on the procedure for performing the policy evaluation step for such infinite state space systems, and specializing it to the problem of solving the optimal dynamic MPL control problem.

Consider a fixed policy π, and let P^π denote the probability transition matrix:

$$P^\pi = \begin{bmatrix} L_0 & F_0 & 0 & 0 & 0 & \cdots \\ B_0 & L & F & 0 & 0 & \cdots \\ 0 & B & L & F & 0 & \cdots \\ & \vdots & & & \vdots & \end{bmatrix}$$

Let $\mathbf{h}_0$ be the vector of differential costs for the 0th (non-repeating) level, $\mathbf{h}_i$ ($i \geq 1$) be the differential cost vector for the ith (repeating) level of the state space, and γ be the average cost under policy π. Denote by R the rate matrix (for the embedded chain) which is the least non-negative solution to:

$$R = F + RL + R^2B$$

Let G be the solution to the following equation:

$$G = B + LG + FG^2$$

We now note that G and R have the following probabilistic interpretations [18]: by conditioning on the first transition, it is easy to see that $G(j,k)$ denotes the conditional probability that the chain eventually reaches level $i-1$ and the state it enters is $(i-1,k)$, given the chain starts in state (i,j). Similarly, conditioning on the last transition before visiting $(i+1,k)$, one can see that the entry $R(j,k)$ represents the mean number of visits to state $(i+1,k)$ until it first enters level i again, given that the chain starts in state (i,j). Let J be given by:

$$J = L + FG$$

Then $J(j,k)$ represents the conditional probability that the chain enters level i again before entering level $i-1$, and that the state it enters is (i,k), given the chain starts in state (i,j). We can now write the differential cost of some state (i,j) for $i \geq 2$ as

$$\begin{aligned} h(i,j) &= c(i,j)\tau(i,j) - \gamma\tau(i,j) + \sum_k B(j,k)h(i-1,k) \\ &\quad + \sum_k J(j,k)h(i,k) \\ &\quad + \sum_{m=1}^{\infty}\sum_k R^m(j,k)\left[c(i+m,k)\tau(i+m,k) - \gamma\tau(i+m,k)\right] \end{aligned}$$

or,

$$\begin{aligned} \mathbf{h}_i &= diag(\boldsymbol{\tau}_i)\mathbf{c}_i - \gamma.\boldsymbol{\tau}_i + B\mathbf{h}_{i-1} + J\mathbf{h}_i \\ &\quad + \sum_{m=1}^{\infty} R^m(diag(\boldsymbol{\tau}_{i+m})\mathbf{c}_{i+m} - \gamma\boldsymbol{\tau}_{i+m}) \quad \ldots i \geq 2 \end{aligned} \tag{15}$$

where $\mathbf{c}_i$ is the column vector of cost per unit of time for states in level i, and $\boldsymbol{\tau}_i$ is the column vector of mean residence time for states in level i. Thus,

$$\begin{aligned} \mathbf{h}_i = (I-J)^{-1}\Big(&diag(\boldsymbol{\tau}_i)\mathbf{c}_i - \gamma\boldsymbol{\tau}_i + B\mathbf{h}_{i-1} \\ &+ \sum_{m=1}^{\infty} R^m(diag(\boldsymbol{\tau}_{i+m})\mathbf{c}_{i+m} - \gamma\boldsymbol{\tau}_{i+m})\Big) \end{aligned}$$

Thus, we can express $\mathbf{h}_2$ in terms of $\mathbf{h}_1$ and solve for $\mathbf{h}_0$ and $\mathbf{h}_1$. We can then obtain subsequent cost vectors as needed while performing the policy improvement step.

We now address the problem of evaluating a dynamic MPL control policy, π. Let K^+ denote the maximum MPL used by policy π and let Q^* denote the queue length beyond which policy π uses the MPL K^* (see Figure 6). For computational reasons we restrict Q^* to be at most 50. As stated earlier, in our case, the matrix B is of rank 1. Specifically, we can write $B = \boldsymbol{\nu}\cdot\boldsymbol{\alpha}$, where $\boldsymbol{\nu} = \boldsymbol{e}_1$ (the column vector with first entry 1, and rest 0), and $\boldsymbol{\alpha} = [(1-q)\ q\ 0\ \ldots\ 0]$. Therefore, in our case the matrices G and R have an explicit solution [20]:

$$\begin{aligned} G &= \boldsymbol{e}\cdot\boldsymbol{\alpha} \\ R &= F(I - L - F\boldsymbol{e}\boldsymbol{\alpha})^{-1} \end{aligned}$$

where $\boldsymbol{e}$ is the column vector of all 1s.

Denote by $\mathbf{s_i}$ the state vector for level i, $i \geq 1$:

$$\mathbf{s}_i = \begin{bmatrix} (Q^*+i-1, K^*-1, ?) \\ (Q^*+i-1, K^*) \\ \vdots \\ (Q^*+i-1, K^+) \end{bmatrix}$$

with the cost and mean residence time vectors given by $\mathbf{c}_i = (Q^*+i-1)\cdot\boldsymbol{e} + \mathbf{K}$ and

$$\boldsymbol{\tau}_i = \begin{bmatrix} 0 \\ \frac{1}{\lambda+q\cdot\mu(K^*)} \\ \vdots \\ \frac{1}{\lambda+q\cdot\mu(K^+)} \end{bmatrix} = \boldsymbol{\tau}, \qquad \mathbf{K} = \begin{bmatrix} K^*-1 \\ K^* \\ \vdots \\ K^+ \end{bmatrix}$$

We can thus simplify (15) to:

$$\begin{aligned} \mathbf{h}_i &= diag(\boldsymbol{\tau})\mathbf{c}_i - \gamma\boldsymbol{\tau} + B\mathbf{h}_{i-1} + J\mathbf{h}_i \\ &\quad + \sum_{m=1}^{\infty} R^m(diag(\boldsymbol{\tau})\mathbf{c}_{i+m} - \gamma\boldsymbol{\tau}) \\ &= B\mathbf{h}_{i-1} + J\mathbf{h}_i + \Big[(I-R)^{-1}\left((Q^*+i-2-\gamma).\boldsymbol{\tau}\right. \\ &\quad \left. + diag(\boldsymbol{\tau})\cdot\mathbf{K}\right) + \left((I-R)^{-1}\right)^2\boldsymbol{\tau}\Big] \end{aligned} \tag{16}$$

or,

$$\begin{aligned} \mathbf{h}_i = (I-J)^{-1}\Big\{&B\mathbf{h}_{i-1} + \Big[(I-R)^{-1}\left((Q^*+i-2-\gamma).\boldsymbol{\tau}\right. \\ &\left. + diag(\boldsymbol{\tau})\mathbf{K}\right) + \left((I-R)^{-1}\right)^2\boldsymbol{\tau}\Big]\Big\} \end{aligned} \tag{17}$$

Thus, the solution of our system is given by the following system of linear equations for $\mathbf{h}_0, \mathbf{h}_1, \gamma$:

$$\mathbf{h}_0 = diag(\boldsymbol{\tau}_0)\mathbf{c}_0 - \gamma\boldsymbol{\tau}_0 + L_0\mathbf{h}_0 + F_0\mathbf{h_1} \tag{18}$$

$$\begin{aligned} \mathbf{h}_1 &= diag(\boldsymbol{\tau})\mathbf{c}_1 - \gamma\boldsymbol{\tau} + B_0\mathbf{h}_0 + L\mathbf{h}_1 + F\mathbf{h_2} \\ &= diag(\boldsymbol{\tau})\mathbf{c}_1 - \gamma(I + F(I-J)^{-1}(I-R)^{-1})\boldsymbol{\tau} + B_0\mathbf{h}_0 \\ &\quad + (L + F(I-J)^{-1}B)\mathbf{h}_1 + F(I-J)^{-1}\Big\{\Big[(I-R)^{-1}\left(Q^*.\boldsymbol{\tau}\right. \\ &\quad \left. + diag(\boldsymbol{\tau})\mathbf{K}\right) + \left((I-R)^{-1}\right)^2\boldsymbol{\tau}\Big]\Big\} \end{aligned} \tag{19}$$

and the additional constraint $h(0,0) = 0$.

In the method of policy iteration, the policy evaluation and policy improvement steps are repeated until two policies with the same cost are obtained. In the experiments presented in this paper, we stopped when the relative improvement between consecutive policies was below 0.01%, which took at most 7 iterations in each case (less than 30 seconds on a Pentium 4 CPU with 1 GB of memory).