Heavy-Traffic Revenue Maximization in Parallel Multiclass Queues

Jonatha Anselmi[1] and Giuliano Casale[2]
[1]Basque Center for Applied Mathematics (BCAM)
Al. de Mazarredo 14, 48009 Bilbao, Basque Country, Spain
[2]Imperial College London, Department of Computing
180 Queens Gate, SW7 2AZ, London, U.K.
anselmi@bcamath.org, g.casale@imperial.ac.uk

Abstract

Motivated by revenue maximization in server farms with admission control, we investigate optimal scheduling in parallel processor-sharing queues. Incoming customers are distinguished in multiple classes and we define revenue as a weighted sum of class throughputs. Under these assumptions, we describe a heavy-traffic limit for the revenue maximization problem and study the asymptotic properties of the optimization model as the number of clients increases. Our main result is a simple heuristic that is able to provide tight guarantees on the optimality gap of its solutions. In the general case with M queues and R classes, we prove that our heuristic is $(1 + \frac{1}{M-1})$-competitive in heavy-traffic. Experimental results indicate that the proposed heuristic is remarkably accurate, despite its negligible computational costs, both in random instances and using service rates of a web application measured on multiple cloud deployments.[1]

1 Introduction

The problem of scheduling a set of customers among a set of parallel queues is classic in queueing theory and operations research. Despite the vast literature on this subject, finding *efficiently* optimal (or near-optimal) solutions remains a difficult problem in several cases of practical interest. We focus in particular on closed, multiclass queueing networks in the sense of Kelly [23] or Baskett et al. [8], and we are interested in the problem of finding a *probabilistic policy* that maximizes the *revenue* of a network of parallel servers. A probabilistic policy (or random splitting) is a routing policy where customers of each class are sent to servers according to a fixed i.i.d. probability law, and the revenue is defined as a weighted summation of the class throughputs. In this setting, approximations and numerical optimization methods have been proposed in the past by many authors, e.g., [17, 24]. However, existing methods either lack guarantees on the optimality gap of their solutions or their computational costs do not scale well in practice, as we illustrate in this paper. Motivated by this, we develop a novel heavy-traffic analysis to assign job routing probabilities to servers. Our solution features a guaranteed optimality gap and it is coupled with rigorous characterization results.

Several works have addressed the problem of finding a probabilistic policy that optimizes some performance criterion. For *open* systems, i.e., systems with arrivals from the external world, this problem is captured by an optimization problem that is in general non-convex [13,18,21]. We here instead focus on *closed* systems, where a fixed population of customers cyclically issues requests to the servers; in these models, the problem becomes even more difficult because, besides remaining non-convex, a single evaluation of the objective function is often much more expensive [17, 24, 38]. This holds because non-iterative expressions for the stationary performance indexes are lacking for multiclass closed models, thus most formulations require to use gradient-hill methods in combination with some model evaluation technique [10,17,24], e.g., approximate mean-value analysis (MVA) [15, 33, 43]. These formulations are popular in applications [5, 16, 31, 39] but, as

[1]Research partially supported by grant MTM2010-17405 (Ministerio de Ciencia e Innovación, Spain) and grant SA-2012/00331 of the Department of Industry, Innovation, Trade and Tourism (Basque Government) which sponsored a visit of the second author to BCAM in November 2012. The research leading to these results has also received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement no. 318484.

we show in the paper, they often fail to converge to a local optimum at the short timescales of minutes at which online capacity management systems operate today. This implies that practitioners can be forced to work with solutions that may be quite far from the global (or even a local) optimum. We tackle these issues by developing a simple heuristic, which provides a closed-form expression for the routing probabilities and tight optimality guarantees in heavy-traffic. Our heuristic is based on the simple idea that each station gives priority to the class(es) that maximizes the revenue achievable by that queue. This scheme is reminiscent of the $c\mu$ rule[2] in the scheduling of customers in a single-server queue [19], even though that is a different scenario from the one considered here. Perhaps surprisingly, our heuristic features an optimality gap that decreases as the number of servers in the model increases, thus making it more accurate as the model becomes larger. We also observe that its relative revenue compared to existing techniques improves as the number of workload classes grows. Experimental results indicate that this heuristic is remarkably accurate and often outperforms existing methods used in applications. Furthermore, our heuristic is robust in the sense that it does not depend on the mix of customers, which in practice changes over time, and it is fast enough for external use in complex optimization models that need to evaluate thousands of subproblems. Finally, we include in the paper characterization results in heavy-traffic that clarify the nature of the revenue maximization problem in this regime. This bears similarities with existing approaches based on approximate MVA.

Summarizing, our main contributions are

- A novel heuristic for assignment of routing probabilities in multiclass networks of parallel processor sharing queues with a tight optimality gap in heavy-traffic;
- In addition to the heuristic, we develop rigorous characterization results for the heavy-traffic limit of the revenue maximization problem;
- A large experimental validation that demonstrates the limitations of current gradient-hill based approaches used in the literature and numerical evidence that our heuristic performs better than these methods, particularly on large instances, and even outside its underpinning theoretical assumptions;
- Section 6 illustrates our method on models parameterized using execution traces of web applications deployed on multiple clouds.

This paper is organized as follows. Section 2 overviews related work and applications. Section 3 presents the reference model of our analysis and the problem statement. Our technical results are given in Section 4 and are evaluated in Section 5 by numerical tests. A validation on cloud traces is developed in Section 6. Conclusions are given in Section 7. Finally, the appendix includes the proof of Theorem 2.

2 Related work

There is a wide literature on the problem of scheduling a set of customers among a set of parallel queues. The goal is to design a scheduling policy that a central dispatcher should implement to maximize some performance criterion. There are two main types of scheduling policies [1]: *closed-loop* policies, where the central dispatcher knows the state (e.g., number of customers) of each queue, or *open-loop* policies, where the central dispatcher is agnostic of states. We focus on the latter policies, which incur less overhead and are easier to implement. Within open-loop policies, we restrict the focus on policies where customers are sent to queues according to fixed probabilities (probabilistic policies), which is a case that received large attention in the literature [13, 17, 22, 24, 34, 36, 38]. This choice enhances tractability, as finding the best non-probabilistic open-loop policy is NP-complete [7], and is also motivated by the fact that there exist cases where the performance achieved by an optimal probabilistic policy is very close to the performance achieved by the optimal open-loop policy [4].

Optimization problems that aim at finding an optimal probabilistic policy are convex only in some particular cases. For instance, in open systems, convexity appears in cases where one considers M/M/1 queues [9], while in closed systems convexity appears in single-class product-form networks [22, 24, 36]. Of course, convexity is an important property that allows one to design algorithms that converge to an optimal

[2]In a single queue, the $c\mu$ rule gives priority to the customer that maximize $c_i\mu_i$ where c_i is the holding cost of customer i and μ_i^{-1} is its service time.

solution quite fast [14]. In our closed multiclass queueing network, the revenue maximization problem is non-convex, as also discussed in [24].

In closed multiclass queueing networks, a number of works have characterized the structure of the solution of the revenue maximization problem [17, 22, 24, 38]. The main result is the vertex allocation theorem [38] and its generalizations [17, 22], which states that there exists an optimal policy where *each* customer follows a deterministic path in the network. Furthermore, for *a given* customer, such policy can be given by the solution of a linear program [22], though its parameters require the solution of two queueing network models for each node of the network, which can be quite expensive. This linear program, in turn, can be used to develop an algorithm to compute a Nash equilibrium. Our context is different from [22] because we are interested in optimal probabilistic policies for *all* customers simultaneously: from a game-theoretic standpoint, we are interested in the "social-optimum" problem rather than finding a Nash equilibrium.

In terms of applications, the development of optimal probabilistic policies in closed systems is a problem that appears in communication networks with applications to computer systems and urban/road networks among others [36]. Revenue-based systems are also becoming increasingly important in software systems, spanning from middlewares for web-service composition to software-as-a-service applications which differentiate performance across customers depending on their subscription model [20, 26, 42].

3 Model

3.1 Notation and Assumptions

We consider closed multiclass queueing networks in the sense of Kelly [23] and Baskett et al. [8]. We assume that the network is composed of M processor-sharing queues (or servers) working in parallel. There is a constant number of jobs that circulate in the network, a typical assumption to represent admission control (i.e., concurrency limits) in servers [39]. Customers are partitioned in R classes and N_r denotes the number of customers of class r. We let $N = (N_1, \ldots, N_R)$ denote the population vector. Queues are indexed by $i = 1, \ldots, M$; classes by $r, s = 1, \ldots, R$. We do not model customers' "think times" (or infinite-server queues) because their impact is negligible in our heavy-traffic limit (described later). Also, we do not model the dispatcher because we assume that congestion at this node is negligible.

Upon completion of a class-r request at each queue, each class-r customer issues a new request that is routed to queue i with probability p_{ir}. Service times are i.i.d. Coxian with mean μ_{ir}^{-1} and arbitrary higher-order moments. In the following, we let $p \in \mathbb{R}_+^{MR}$ denote the routing probability matrix, thus $\sum_i p_{ir} = 1$, for all r. We let $H_r \subseteq \{1, \ldots, M\}$ be the set of processor-sharing queues that class-r customers is allowed to visit, i.e., $p_{ir} = 0$ for all $i \notin H_r$, for all r. We assume $\mu_{ir} > 0$ for all i and r and that each processor-sharing queue is visited by at least one class, i.e., $i \in \bigcup_r H_r$ for all i.

Let $n \in \mathcal{S} \stackrel{\text{def}}{=} \{n \in \mathbb{Z}_+^{MR} : \sum_i n_{ir} = N_r, \forall r, \text{ and } n_{ir} > 0 \text{ if and only if } p_{ir} > 0, \forall i, r\}$ denote a *state* of the network, i.e. an allocation of customers among the queues. Under the above assumptions, the stationary probability of being in state n, denoted by $\pi_{N,p}(n)$, assumes a simple product-form expression given by [8, 23]

$$\pi_{N,p}(n) = \frac{1}{G(N)} \prod_i n_i! \prod_{r:i \in H_r} \frac{(p_{ir}\mu_{ir}^{-1})^{n_{ir}}}{n_{ir}!}, \qquad n \in \mathcal{S} \tag{1}$$

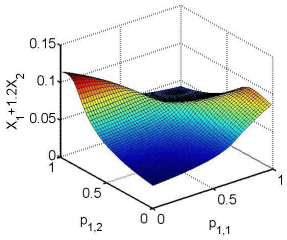
where $n_i \stackrel{\text{def}}{=} \sum_{r:i \in H_r} n_{ir}$ and

$$G(N) \stackrel{\text{def}}{=} \sum_{k \in \mathcal{S}} \prod_i k_i! \prod_{r:i \in H_r} \frac{(p_{ir}\mu_{ir}^{-1})^{k_{ir}}}{k_{ir}!} \tag{2}$$

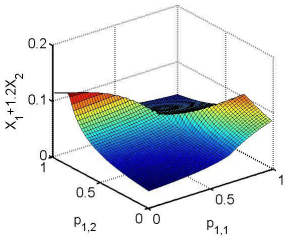
is a normalizing constant.

Denote by $X_r(N, p)$ the *throughput* for class-r customers, that is the aggregate mean departure rate from all the queues. For class-r customers, it can be defined as follows

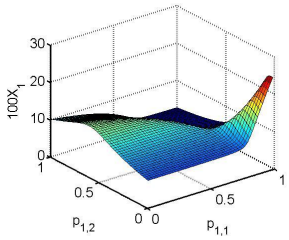
$$X_r(N, p) \stackrel{\text{def}}{=} \sum_{i \in H_r} X_{ir}(N, p) \tag{3}$$



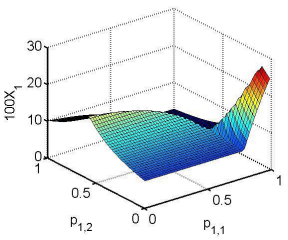
(a) $c_1 = 1$, $c_2 = 1.2$, $\mu_{11}^{-1} = 10$, $\mu_{12}^{-1} = 63$, $\mu_{21}^{-1} = 15$, $\mu_{22}^{-1} = 86$, $N_1 = 10$, $N_2 = 25$.



(b) Plot of (a) in heavy traffic ($N_1 = 1000$, $N_2 = 2500$) .



(c) $c_1 = 100$, $c_2 = 0$, $\mu_{11}^{-1} = 10$, $\mu_{12}^{-1} = 6$, $\mu_{21}^{-1} = 4$, $\mu_{22}^{-1} = 16$, $N_1 = 10$, $N_2 = 25$.



(d) Plot of (c) in heavy traffic ($N_1 = 1000$, $N_2 = 2500$).

Figure 1: Plot of $\Pi(N, p)$ for varying routing matrices p. In all figures, $M = R = 2$.

where

$$X_{ir}(N,p) \stackrel{\text{def}}{=} \sum_{n \in \mathcal{S}: n_{ir}>0} \mu_{ir} \frac{n_{ir}}{\sum_s n_{is}} \pi_{N,p}(n) \tag{4}$$

represents the throughput of class-r customers from queue i under processor-sharing scheduling.

3.2 Revenue Maximization Problem

With respect to positive weights $c \in \mathbb{R}^M$, we define the system *revenue* as

$$\Pi(N,p) \stackrel{\text{def}}{=} \sum_r c_r X_r(N,p). \tag{5}$$

It is natural to consider the problem of finding the routing probabilities that maximize[3] $\Pi(N,p)$. Formally, we want to solve the following optimization problem

$$\begin{array}{rll} \Pi^*(N) \stackrel{\text{def}}{=} \max\limits_{p \geq 0} & \Pi(N,p) & \\ \text{s.t.:} & \sum_i p_{ir} = 1, & \forall r \\ & p_{ir} = 0, & \forall i \notin H_r, r. \end{array} \tag{6}$$

Figure 1 plots the revenue function $\Pi(N,p)$ by varying p_{11} and p_{12} with respect to two models with two processor-sharing queues and two classes (naturally, $p_{21} = 1 - p_{11}$ and $p_{22} = 1 - p_{12}$). The figures provide intuition of the challenges of the problem. Multiple local optima exist and the gradients can be very different on the surfaces, making the underlying optimization problems neither convex nor concave and in our experience quite sensitive numerically. Evidently, for larger number of queues and classes the problem becomes much more challenging since different local optima can yield very different revenues, as seen for example in Figure 1(c). Figure 1(a) suggests the idea that starting from a non-optimal point (e.g., (0,0)), any solver can have little information to discriminate the search direction that will lead to the global optimum. Furthermore, in Figure 1(c), one can see by eye that if a solver starts in (0,0) and follows the gradient of $\Pi(N,p)$, then it eventually converges to the local maximum on the left side, instead of the global maximum on the right side.

We have also investigated in Figure 1(b) and 1(d) the same models in heavy-load with a scaling factor of 100 for each class population. We have found that the values of the throughputs were slightly larger, but the shapes of the plots were qualitatively similar. This provides intuition that for these models the location of the local optimum solutions may *not* be significantly altered by the heavy-traffic limit assumption.

Other constraints may be added to the optimization model (6). In particular, we may require that the utilization of each queue i, i.e., the proportion of time in which queues are busy processing customers, is less

[3]When $c = 1$, the optimization problem (6) is equivalent to the problem of minimizing the network *response time*, i.e., the mean time that customers take to complete a cycle in the network. In fact, using Little's law, the response time of the network can be defined as $\frac{\sum_r N_r}{\sum_r X_r(N,p)}$. Since $\sum_r N_r$ is a constant, it is clear that minimizing such quantity is equivalent to maximizing $\sum_r X_r(N,p)$.

than a given constant $U^{\max}$, which is useful to avoid excessive contention at each server. Using Little's law, it is sufficient to add

$$\sum_r \frac{p_{ir}}{\mu_{ir}} X_r(N,p) \leq U^{\max}, \tag{7}$$

for all i, to the constraints of (6). However, without loss of generality, constraints of the form (7) can be eliminated by making the substitution $\mu_{ir} \leftarrow \mu_{ir} U^{\max}$, for all i, r, and interpreting $X_r(N,p)$ as the class-r throughput of a queueing network with the scaled service rates, we have $\sum_r \frac{p_{ir}}{\mu_{ir} U^{\max}} X_r(N,p) \leq 1$. Since this constraint is true for any p, it is redundant. Thus, the optimal solution of the scaled model is optimal also for the initial, unscaled, model.

4 Heavy-Traffic Approximations

In this section, we study the revenue maximization problem (6) in the limiting regime where the number of customers grows to infinity and remains proportionate in each class, i.e., N is multiplied by the positive integer k and we let $k \to \infty$. This limiting regime has been used in several works, see [2, 3, 6, 30, 32, 40] and references therein; we refer to this regime as *heavy-traffic*. In this heavy-traffic limit, we provide two results:

- In Section 4.1, we present an efficient heuristic for solving the revenue maximization problem and prove that in heavy traffic it is $\left(1 + \frac{1}{M-1}\right)$-competitive. Since in practice the number of servers M is often of the order of tens in applications, this heuristic is nearly optimal, provided that the load is large. We show in the experimental result section that it is also surprisingly effective under normal load conditions.
- In Section 4.2, we develop the heavy-traffic limit of the revenue maximization problem. The limit maximization problem is not concave, which is to be expected because the revenue function is not concave in the routing probabilities for any load conditions; see Figure 1.

Importantly, the characterization in Section 4.2 explains *what* optimization problem the heuristic is trying to solve, i.e., the asymptotic limit of the revenue maximization problem (6). From this, we observe clear structural similarities with known non-asymptotic formulations, used in applications, which are helpful to understand the experimental results in Section 5.

4.1 A Heuristic with Optimality Guarantees

The next theorem develops an efficient heuristic that is able to provide tight guarantees on the optimality gap of its solutions, and it is our main result.

Theorem 1. *Let queues be ordered such that*

$$\max_{r:1\in H_r} c_r\mu_{1r} \leq \max_{r:2\in H_r} c_r\mu_{2r} \leq \cdots \leq \max_{r:M\in H_r} c_r\mu_{Mr}. \tag{8}$$

Let $r^*(i) \stackrel{\text{def}}{=} \arg\max_{r:i\in H_r} c_r\mu_{ir}$ *and assume that* $r^*(i)$ *is a singleton, for each queue* i*. For* $m \in \{1,\ldots,M\}$*, let* $p^{(m)} \in \mathbb{R}_+^{MR}$ *be any routing matrix that satisfies*

$$\sum_{i\in H_r} p_{ir}^{(m)} = 1, \qquad \forall r \tag{9a}$$

$$p_{ir^*(i)}^{(m)} = \frac{\mu_{ir^*(i)}}{\sum\limits_{\substack{j=m,\ldots,M:\\ r^*(j)=r^*(i)}} \mu_{jr^*(j)}}, \qquad \forall i = m,\ldots,M \tag{9b}$$

$$p_{ir}^{(m)} = 0, \qquad \forall r \neq r^*(i), i = m,\ldots,M. \tag{9c}$$

Then,

$$\lim_{k\to\infty} \frac{\Pi(kN,p)}{\Pi(kN,p^{(m)})} \leq 1 + \frac{(m-1)}{M-(m-1)}, \qquad \forall p. \tag{10}$$

Proof. Our proof requires the following two propositions. The first bounds $\Pi(N,p)$, while the second gives the asymptotic behavior of the throughput for a given p and was proven in [40] (see also [3]).

Proposition 1.

$$\Pi(N,p) \le \sum_i \max_{r:i\in H_r} c_r \mu_{ir}, \qquad \forall N,p. \tag{11}$$

Proof. Since $\sum_{r:i\in H_r} c_r\mu_{ir}\frac{n_{ir}}{\sum_s n_{is}} \le \max_{r:i\in H_r} c_r\mu_{ir}$, when $\sum_s n_{is} > 0$, using (4) we have

$$\sum_{r:i\in H_r} c_r X_{ir}(N,p) \le \max_{r:i\in H_r} c_r\mu_{ir}. \tag{12}$$

Using (3) and summing over i we obtain (11). □

Proposition 2 ([40])**.** *For all r, $\lim_{k\to\infty} X_r(kN,p) = X_r^*(p)$ where $X^*(p) \stackrel{\text{def}}{=} (X_1^*(p),\ldots,X_R^*(p))$ is the unique solution of the strictly-concave optimization problem*

$$\max_{X\ge 0} \sum_r N_r \log X_r \tag{13a}$$

$$\text{s.t.: } \sum_r \tfrac{p_{ir}}{\mu_{ir}} X_r \le 1, \qquad \forall i. \tag{13b}$$

Now, let $\mathcal{R} \stackrel{\text{def}}{=} \{r : \text{ there exists } i \in \{m,\ldots,M\} \text{ such that } r^*(i) = r\}$, and let $\mathcal{R}^c \stackrel{\text{def}}{=} \{1,\ldots,R\} \setminus \mathcal{R}$. Using (13) and (9), $X^*(p^{(m)})$ is uniquely given by the optimizer of

$$\max_{X\ge 0} \sum_r N_r \log X_r \tag{14a}$$

$$\text{s.t.: } X_{r^*(i)} \le \sum_{\substack{j=m,\ldots,M:\\ r^*(j)=r^*(i)}} \mu_{jr^*(j)}, \qquad \forall i = m,\ldots,M \tag{14b}$$

$$\sum_{r\in\mathcal{R}^c: i\in H_r} \tfrac{p_{ir}^{(m)}}{\mu_{ir}} X_r \le 1, \qquad \forall i = 1,\ldots,m-1. \tag{14c}$$

For any $i \ge m$, the monotonicity of the objective function with respect to $X_{r^*(i)}$ and given that $X_{r^*(i)}$ does not appear, by definition of $p^{(m)}$, in constraints (14c) imply that (14b) holds with equality in the optimal solution, that is

$$X_r^*(p^{(m)}) = \sum_{\substack{j=m,\ldots,M:\\ r^*(j)=r}} \mu_{jr^*(j)}, \qquad \forall r \in \mathcal{R}. \tag{15}$$

Therefore, for $\Pi^*(p^{(m)})$ we have

$$\Pi^*(p^{(m)}) = \sum_r c_r X_r^*(p^{(m)}) \ge \sum_{r\in\mathcal{R}} c_r \sum_{\substack{j=m,\ldots,M:\\ r^*(j)=r}} \mu_{jr^*(j)} = \sum_{i=m}^{M} c_{r^*(i)}\mu_{ir^*(i)} = \sum_{i=m}^{M} \max_{r:i\in H_r} c_r\mu_{ir}. \tag{16}$$

Now, by Proposition 1, $\Pi^*(p) \le \sum_i \max_r c_r\mu_{ir}$ and we obtain

$$\frac{\Pi^*(p)}{\Pi^*(p^{(m)})} \le \frac{\sum_i \max_{r:i\in H_r} c_r\mu_{ir}}{\sum_{i=m}^{M} \max_{r:i\in H_r} c_r\mu_{ir}} \le 1 + \frac{(m-1)\max_{r:i\in H_r} c_r\mu_{mr}}{(M-m+1)\max_{r:i\in H_r} c_r\mu_{mr}} = 1 + \frac{m-1}{M-m+1} \tag{17}$$

where the last inequality uses the ordering (8). □

4.1.1 Discussion

Some remarks are needed to clarify the application of Theorem 1:

i) The routing matrix $p^{(m)}$ ensures that queue i, for each $i = m, \ldots, M$, is only visited by the class r that maximizes $c_r \mu_{ir}$ and that the load among all queues associated with a given class is perfectly balanced. This is enough to guarantee that $\lim_{k\to\infty} \Pi(kN, p^{(m)})$ is $\left(1 + \frac{m-1}{M-m+1}\right)$-competitive.

ii) If $m = 1$, $p^{(m)}$ is indeed an asymptotically-optimal solution (actually, one can easily see that it is the unique optimal solution). However, the problem here is that it may be not possible to construct a routing matrix $p^{(1)}$; that is, the linear system (9) may have no solution in $\mathbb{R}_+^{MR}$. In fact, given the constraint that all queues must be visited by only one class, such $p^{(1)}$ cannot be constructed if $R > M$, which is an important case in practice. More generally, $p^{(1)}$ can be constructed if and only if $\bigcup_i r^*(i) = \{1, \ldots, R\}$.

iii) If $m \geq 2$, $p^{(m)}$ can always be constructed. In particular, the linear system (9) has a unique solution if $m = 2$ and infinite solutions if $m > 2$. Furthermore, if $m = 2$, then $p^{(2)}$ allocates exactly one class to each queue $i = 2, \ldots, M$ and puts all the other classes (regardless of their number) in queue 1. This allocation is $\left(1 + \frac{1}{M-1}\right)$-competitive, which means that it is almost optimal (asymptotically) because M is large in practice.

iv) While $p^{(2)}$ is the allocation that provides the smallest guarantee of optimality gap, it has the drawback that several classes may be allocated in one queue, namely, queue 1. In practice, this may not be a good solution for aspects such as reliability, or scalability in terms of memory requirements. However, one can choose $m > 2$ large enough and force that classes are distributed among multiple queues at a desired level while guaranteeing a level of optimality gap.

v) An important property of our heuristic is that $p^{(m)}$ does not depend on the population vector N. This makes our heuristic robust to changes of N, which in practice may vary over time. For example, in [37] it is shown that web applications have non-stationary request mixes, thus requiring the estimation of model parameters at short timescales. Our heuristic is insensitive to both N and to the mix of customers and therefore would avoid the overhead of monitoring and continuous re-estimation of the population mix.

4.2 Revenue Maximization in Heavy-Traffic

The heuristic presented in Section 4.1 depends on Proposition 2, which says that the asymptotic throughputs of the closed multiclass networks under investigation is given by the solution of a strictly concave optimization. This result, proven recently in [40] and [3] under different perspectives, is known to have relationships with the proportionally-fair allocation achieved in communication networks with congestion control, and, to the best of our knowledge, it has not been applied to revenue maximization in the sense of (6). By investigating the KKT conditions of the resulting formulation, it is possible to establish a connection between this asymptotic formulation and the most common formulation of the revenue maximization problem based on the Bard-Schweitzer approximate MVA algorithm [33], which probably is the approximation for closed models that is most popular among practitioners [29].

In this section, we study the heavy-traffic limit of (6). Our main result is Theorem 2, which needs the following technical assumption.

Assumption 1 (Throughput monotonicity)**.** *For each class r and routing probability matrix p, there exists a k^* such that $X_r(kN, p)$ is either non-increasing or non-decreasing in k for all $k > k^*$.*

Assumption 1 was proved in the case of single-class queueing networks [35]. In multiclass networks, it is known that the class throughputs are *not* monotone in general when the population is kN and k grows [6, Figure 2]. However, they are believed to be monotone *eventually* [6], that is for all $k > k^*$, for a large enough k^*. From a technical standpoint, we need this assumption to establish uniform convergence of the throughputs over the set of routing probabilities, which allows exchanging the limit and maximum

operators; see Lemma 2 in the appendix. Under this assumption, we can prove the following result, which gives the heavy-traffic limit of $\Pi^*(N)$.

Theorem 2. *Let Assumption 1 hold. Then,*

$$\lim_{k\to\infty} \Pi^*(kN) = \max_{X\in\mathbb{R}_+^{MR},\, L\in\mathbb{R}_+^{M}} \sum_r c_r \sum_{i\in H_r} X_{ir} \tag{18a}$$

$$\text{s.t.:} \sum_{i\in H_r} \tfrac{X_{ir}}{\mu_{ir}} L_i = N_r,\ \forall r \tag{18b}$$

$$\sum_{r:i\in H_r} \tfrac{X_{ir}}{\mu_{ir}} = 1,\ \forall i. \tag{18c}$$

Furthermore, let (X^*, L^*) *be an optimizer of* (18) *and define* $p^*_{ir} = \frac{X^*_{ir}}{\sum_{j\in H_r} X^*_{jr}}$*, for all* i *and* r*. Then,*

$$\lim_{k\to\infty} \Pi^*(kN) = \lim_{k\to\infty} \Pi(kN, p^*). \tag{19}$$

The proof of this theorem is given in the appendix. We remark that in (18), the variables X_{ir} can each be interpreted as the throughput of class-r on queue i, as in (4), and variables L_i can each be interpreted as the mean number of customers at queue i normalized by k (it is not difficult to see that (18b) and (18c) imply $\sum_i L_i = \sum_r N_r$).

Optimization problem (18) does not fall in a category of optimization problems that are known to be solvable by efficient polynomial-time algorithms, to the best of our knowledge. This is only due to constraint (18b), which is bilinear. Bilinear programs are known to be NP-hard, in general [11]. However, if both the number of queues and classes is not too large, we have found existing interior-point methods for convex programming (see [14]) to converge to a local optimum quite fast.

4.2.1 Connection to Bard-Schweitzer Formulation

As observed earlier, we now wish to establish a connection between (18) and existing formulations used in applications for revenue maximization. Recall that, for a closed multiclass queueing network, the Bard-Schweitzer approximation is a non-linear system of equations used to approximate performance indexes in constant time with respect to the total customer population [33]. One can readily reformulate (6) in such a way to use the Bard-Schweitzer approximation for the computation of the performance indexes. That is

$$\Pi^{bs}(N) = \max_{X\in\mathbb{R}_+^{MR},\, n\in\mathbb{R}_+^{MR}} \sum_r c_r \sum_{i\in H_r} X_{ir} \tag{20a}$$

$$s.t.: \quad \tfrac{X_{ir}}{\mu_{ir}}(1 + \textstyle\sum_s n_{is} - \frac{n_{ir}}{N_r}) = n_{ir},\ \forall r, i\in H_r \tag{20b}$$

$$\sum_{r:i\in H_r} \tfrac{X_{ir}}{\mu_{ir}} \le 1,\ \forall i. \tag{20c}$$

$$\sum_{i\in H_r} n_{ir} = N_r,\ \forall r \tag{20d}$$

While formulations (18) and (20) are different, it is interesting to notice that (20b) converges to (18b) in the heavy-traffic limit, thus the feasible regions bear some similarities. This is confirmed by our experimental tests, where we illustrate numerically that (20) often describes well the heavy-traffic limit in (18).

5 Numerical Results

We have performed a series of experiments to evaluate the revenue and time requirements achieved by the proposed heavy-traffic approximations. In particular, our goal is to understand the limit of applicability of these approximations when the network operates in a regime with *finite* populations and to compare different approaches. Based on empirical evidence, our main conclusion is that the heuristic proposed in Theorem 1 provides a revenue that is competitive with existing optimization approaches for small models, and outperforms existing methods for medium and large models.

5.1 Comparative Analysis

5.1.1 Instances

Initially, we have considered thousands of random models generated by varying the number of stations, classes and jobs within given ranges and by randomly generating service rates and class revenues in each experiment. The number of stations and classes are varied in $M, R = 2, 4, 8, 16, 32, 64$; the number of jobs are varied in $N = 10M, 50M$. For each chosen tuple (M, R, N), we have generated 300 models with service rates μ_{ir} and revenues c_r drawn from a uniform distribution in $[1, 100]$. To enhance reproducibility of the results, random values are generated using the gallery command in MATLAB[4]. The class populations N_r are obtained by diving equally the total number of jobs N across classes and rounding up to the nearest larger integer, and we correct the last population so that $\sum_r N_r = N$, provided that this does not yield an infeasible[5] value of N_r.

5.1.2 Solution Techniques

For each random model, we compare the following methods for revenue maximization:

- asy: the heavy-traffic formulation in Theorem 2;
- ut: the heavy-traffic formulation in Theorem 2 without (18b);
- mcc: a linear program obtained by applying McCormick's convex relaxation [28] to (18b);
- fpm: the heavy-traffic formulation in Theorem 2 with (18b) replaced by Whitt's Fixed Point Method equation [41]:

$$\sum_{i \in H_r} \frac{\sum_r \frac{X_{ir}}{\mu_{ir}}}{1 - \sum_r \frac{X_{ir}}{\mu_{ir}}} = N_r, \ \forall r \tag{21}$$

- bs: the heavy-traffic formulation in Theorem 2 with (18b) replaced by the Bard-Schweitzer's fixed-point equation [33]:

$$\sum_{i \in H_r} \frac{X_{ir}}{\mu_{ir}} (1 + \sum_s n_{is} - \frac{n_{ir}}{N_r}) = n_{ir}, \ \forall r, i \tag{22}$$

- bs-grad: the heavy-traffic formulation in Theorem 2 without (18b) and with the throughputs in the objective function computed by the Bard-Schweitzer's fixed-point iteration.
- aql-grad: the heavy-traffic formulation in Theorem 2 without (18b) and with the throughputs in the objective function computed by the AQL fixed-point iteration [43].
- heur-m2: the heuristic in Theorem 1 with $m = 2$.

In all the above formulations, the maximum utilization has been set to $U^{max} = 1$.

A number of comments are required to justify our choice of the above methods. First, the choice of experimenting with different possible replacements for (18b) is motivated by the fact that this is the only non-convex constraint in the formulation of Theorem 2. We have chosen two methods that lead to a looser, but linear, formulation (ut, mcc) and methods that consider different non-convex constraints (fpm, bs). The gradient methods (bs-grad, aql-grad) have been used to establish the competitiveness of approaches that estimate the throughputs X_{ir} by making an external call to an approximate MVA solver. Such approaches are probably the most common in applications [10], together with the basic ut formulation that is equivalent to reasoning based on utilization constraints only. Also, notice that the fpm formulation is immediately related to an open network of $M/GI/1$ processor sharing queues operating with an average population not greater than N_r for each class. Finally, we have focused on validating our heuristic for the case $m = 2$ (heur-m2) since this parametrization provides the best competitive ratio without possible infeasibilities arising in the case $m = 1$, which we already discussed in Section 4.1.1.

[4]Given a tuple (M, R, N), for the jth random experiment, $1 \leq j \leq 300$, we generate model parameters in MATLAB 2012a using the commands $\mu = \text{gallery}('integerdata', 100, [M, R], j)$ for the rate matrix $\mu = [\mu_{ir}]_{M \times R}$, and $c = \text{gallery}('integerdata', 100, [1, R], 10000 * j)$ for the revenue vector $c = [c_r]_{1 \times R}$.

[5]In models with $M = 2$ and $R = 8$ we do not perform this correction since it would lead to negative populations. We just round up to the nearest integer.

Table 1: Mean relative revenue compared to ut. Legend: gap - gap to the best method (in italic).

Instances			Optimization Programs							Heuristic	
M	R	N/M	ut	asy	mcc	fpm	bs	bs-grad	aql-grad	heur-m2	gap
2	2	10	1	1.268	0.983	1.000	1.334	*1.343*	1.342	1.196	*-10.91%*
2	4	10	1	1.360	1.025	1.000	*1.507*	1.486	1.477	1.531	*+1.57%*
2	8	10	1	1.385	1.139	1.002	1.581	*1.582*	1.567	1.675	*+5.88%*
4	2	10	1	1.180	0.988	1.000	1.270	1.258	*1.262*	1.221	*-3.80%*
4	4	10	1	1.310	1.018	1.000	*1.500*	1.430	1.406	1.448	*-3.48%*
4	8	10	1	1.274	1.078	1.007	*1.511*	1.466	1.280	1.488	*-1.54%*
8	2	10	1	1.115	0.994	1.000	*1.199*	1.203	1.108	1.160	*-3.60%*
8	4	10	1	1.183	1.007	0.999	*1.334*	1.296	0.893	1.283	*-3.82%*
8	8	10	1	1.204	0.766	1.008	*1.379*	1.340	0.172	1.341	*-2.75%*
2	2	50	1	*1.335*	0.979	1.000	1.328	1.307	1.288	1.205	*-9.71%*
2	4	50	1	*1.480*	1.023	1.000	1.427	1.445	1.422	1.509	*+1.96%*
2	8	50	1	*1.557*	1.142	1.000	1.519	1.553	1.475	1.695	*+8.82%*
4	2	50	1	1.243	0.988	1.000	*1.253*	1.252	1.190	1.228	*-2.00%*
4	4	50	1	1.435	1.032	1.000	*1.469*	1.438	1.304	1.508	*+2.65%*
4	8	50	1	1.445	1.070	1.009	1.460	*1.468*	0.970	1.554	*+5.89%*
8	2	50	1	1.154	0.994	1.000	*1.189*	1.186	0.929	1.173	*-1.37%*
8	4	50	1	1.270	1.009	0.999	*1.329*	1.310	0.495	1.324	*-0.33%*
8	8	50	1	1.325	1.050	1.009	*1.352*	1.339	0.133	1.402	*+3.69%*

5.1.3 Evaluation Methodology

The above formulations are solved using MATLAB's fmincon interior-point algorithm. For small and medium scale models, this is run on a multi-core server equipped with 8 Xeon CPUs running at 2.53GHz. The large scale models are instead run on a multi-core server with 64 Xeon CPUs running at 2.70GHz. YALMIP is used to generate the runs, but the time results reported refer only to the solver time without the YALMIP overhead [25].

From the solver, we obtain the throughputs X_{ir} which are readily converted into routing probabilities since in a parallel topology $p_{ir} = X_{ir} / \sum_{i \in H_r} X_{ir}$. Notice that the solution $\sum_{i \in H_r} X_{ir} = 0$ is feasible and means that class r is not admitted to service due to its low revenue, we set in this case $N_r = 0$. We use the routing probability values to determine the value of the objective function $\Pi(N, p)$ where the throughputs X_r are computed with AQL, which provides results almost identical to an exact solution algorithm, but at much lower computational costs [43]. Since the final user is interested only in the value of the objective function associated with an actual queueing network, we report only the values computed with AQL. Finally, in order to establish the suitability of each method for online use, we have set a maximum time limit for the execution of the interior-point algorithm to 60 seconds for the small and medium-scale models in Section 5.2 and 600 seconds for large-scale models in Section 5.2. This timeout excludes the time the current iteration needs to terminate[6].

5.2 Small and Medium Scale Random Models

Table 1 and Table 2 report revenue and timing results for the considered methods. Table 1 illustrates the mean relative revenue for all the considered methods compared to the objective function value achieved by ut. The ut formulation is considered as a baseline since it usually provides the worst results due to the poor characterization of the queueing behavior that results from ignoring (18b). The results in the table suggest the following considerations:

- In the experiments, heur-m2 comes out as highly competitive. Its computational costs, shown in Table 2 are negligible, whereas the growth of time requirements for the other methods is already affecting the solution of models with just 4-8 queues. In terms of revenue, the performance of heur-m2 is highly satisfactory: besides becoming more accurate under heavier loads, which is to be expected, relative

[6]For example, if an iteration takes 120s to complete and the timeout is 60s, the algorithm will terminate in the worst case at time 180s. Note that this behavior is forced by fmincon since the user can force a timeout only between iterations.

Table 2: Mean execution time (seconds). Timeout: 60s.

Instances			Optimization Programs							Heuristic
M	R	N/M	ut	asy	mcc	fpm	bs	bs-grad	aql-grad	heur-m2
2	2	10	0.051	0.281	0.119	0.041	0.247	1.355	5.384	0.001
2	4	10	0.070	0.211	0.433	0.054	0.347	4.157	12.067	0.001
2	8	10	0.071	0.175	1.446	0.045	0.272	8.277	25.877	0.001
4	2	10	0.048	0.543	0.364	0.037	0.541	4.197	41.270	0.001
4	4	10	0.069	0.962	9.142	0.048	0.613	6.169	57.046	0.001
4	8	10	0.179	1.516	61.202	0.085	1.322	23.497	timeout	0.001
8	2	10	0.075	1.542	9.522	0.051	1.581	30.693	timeout	0.001
8	4	10	0.133	5.632	59.267	0.074	2.181	27.777	timeout	0.001
8	8	10	0.261	8.553	74.159	0.108	4.514	57.128	timeout	0.001
2	2	50	0.049	0.600	0.120	0.040	0.336	1.896	13.382	0.001
2	4	50	0.064	0.272	0.569	0.050	0.430	3.908	27.159	0.001
2	8	50	0.079	0.333	3.953	0.052	0.523	11.747	48.811	0.001
4	2	50	0.046	0.949	0.517	0.034	1.030	5.297	58.715	0.001
4	4	50	0.070	2.412	3.543	0.047	1.235	7.453	timeout	0.001
4	8	50	0.177	3.973	57.211	0.087	2.385	28.236	timeout	0.001
8	2	50	0.074	2.142	9.859	0.051	3.097	16.605	timeout	0.001
8	4	50	0.133	10.170	58.800	0.073	5.449	38.766	timeout	0.001
8	8	50	0.261	27.940	timeout	0.111	9.854	timeout	timeout	0.001

revenue tends to increase with the number of classes R. Combining this fact with the guaranteed $1 + \frac{1}{M-1}$ optimality gap, which converges to optimality with many queues, we conclude that the overall performance of heur-m2 is very good.

- For the considered model sizes, describing queueing behavior in a non-convex formulation is systematically more effective than considering relaxations (ut, mcc). For ut, the gap to the best method can be up to 55.7%. The mcc results are nearly identical to the ut results. Furthermore, for larger models mcc systematically suffers timeouts on all models due to the large number of constraints generated by applying McCormick's convex relaxation. Also the fpm method appears to be almost identical to ut in terms of revenue, but the structure of the equations seems to help an earlier convergence resulting in faster execution times than those of ut.
- Despite aql-grad is the method that evaluates with the greatest accuracy the queueing network throughputs at each iteration, it should be preferred to the other methods only in a single case ($M = 4$, $R = 2$). As shown in Table 2, aql-grad has the largest execution times across all the methods. Combined with the observations on the performance of ut, which ignores queueing, this seems to suggest that the best results are obtained by striking a balance between accuracy of the queueing representation and iteration speed. This is shown clearly in ($M = 8$, $R = \{4, 8\}$), where the quality of aql-grad degrades due to the heightened time requirements of the iterations.
- The comparison between bs-grad and bs indicates that using blackbox external solvers provides a significant penalty in terms of computational costs. In terms of accuracy the two formulations are theoretically identical, but bs-grad relies on fixed-point iteration for evaluating each model, whereas in bs the Bard-Schweitzer equations are directly handled by the interior point solver. The mean relative revenue results are not conclusive and suggest that at times one method can be slightly better than the other, however overall the advantages of using bs-grad seem little, due to the much larger computational costs than bs.
- In general, bs and asy are the best performing among the optimization programs. Since bs is capable of formulating the problem taking into account for a finite population, it is normally better than asy. However, the asy formulation is quite close to bs in most cases, particularly in heavy-load. This suggests that the asymptotic results in Theorem 2 describe well the behavior of the bs formulation.

Table 3: Mean relative revenue compared to ut. Legend: gap - gap to the best method (in italic).

Instances			Optimiz. Programs			Heuristic	
M	R	N/M	ut	asy	bs	heur-m2	gap
16	16	10	1	1.189	*0.191*	1.280	*+7.63%*
32	32	10	1	1.198	*0.199*	1.283	*+7.10%*
64	64	10	1	*5.423*	3.099	39.256	*+623.74%*
16	16	50	1	1.254	*1.315*	1.340	*+1.86%*
32	32	50	1	*1.255*	0.589	1.326	*+5.65%*
64	64	50	1	*6.593*	0.976	28.912	*+338.50%*

Table 4: Mean Execution Time (seconds). Timeout: 600s.

Instances			Optimization Programs			Heuristic
M	R	N/M	ut	asy	bs	heur-m2
16	16	10	9.62	38.28	306.75	0.01
32	32	10	129.80	timeout	timeout	0.01
64	64	10	timeout	timeout	timeout	0.03
16	16	50	15.45	73.12	291.88	0.01
32	32	50	181.61	timeout	timeout	0.01
64	64	50	timeout	timeout	timeout	0.02

5.3 Large Scale Random Models

In this experimental validation, we focus on investigating the limits of applicability of the best methods of the previous section. We here consider only ut, asy, bs, and heur-m2. Also, to jointly stress the algorithms across all dimensions, we set $M = R$ and increase their common value to generate models with up to 64 queues. We have also attempted larger models with 128 queues, but the solution by means of interior-point methods required more than 32GB of RAM and failed. Numerical results are given in Tables 3 and 4. The main insights are as follows:

- From $M = R = 32$ and above, all optimization programs suffer timeouts since their running times exceed 10 minutes. The results reported are the sub-optimal solutions found before the timeout. Results indicate that only asy and heur-m2 can obtain a solution that is competitive with the one of ut. In fact, bs, which is the slowest of all these methods, either does not progress sufficiently to converge to a good local optimum or terminates early due to numerical problems. This situation happens in all experiments, except for $M = R = 64$ where ut starts to suffer the same problem and therefore the quality of the baseline degrades as well, as visible in the large gaps for these experiments.
- Similarly to the small and medium-sized instances, heur-m2 provides the most efficient solution and, in these experiments, it is always the method with the best revenue. The gap and trends achieved for these models suggest that, even with larger instances, heur-m2 would likely remain the best solution method.

6 Evaluation on Cloud Datasets

In this section, we show the applicability of the proposed heuristic to models parameterized with realistic values of the service rates. We have performed a series of experiments using 7 deployments on 3 cloud providers (GoGrid, Flexiscale, Rackspace) of the demo e-commerce application of Apache OFBiz[7], an open source enterprise resource planning framework. The aim of the experiments is to collect service rates of the different request types processed by the application in a cloud deployment environment. For each cloud provider, we have considered different virtual machine (VM) sizes, as shown in Table 5. The OFBiz demo application is similar to the TPC-W and RUBiS benchmarks, but the framework is more complex since it features several hundreds Java classes and relies on technologies that are common in production systems, such as Groovy dynamic scripting, a template engine, and AJAX.

[7] Apache Ofbiz: http://ofbiz.apache.org/

Table 5: Cloud deployments of the Apache OFBiz web application

VM	*Cloud provider*	*Data Center*	*VM cores*
1	Flexiant	Edinburgh	2
2	Flexiant	Edinburgh	4
3	Flexiant	Edinburgh	6
4	GoGrid	US-East-1	4
5	GoGrid	US-West-1	4
6	Rackspace	Chicago	4
7	Rackspace	Dallas	4

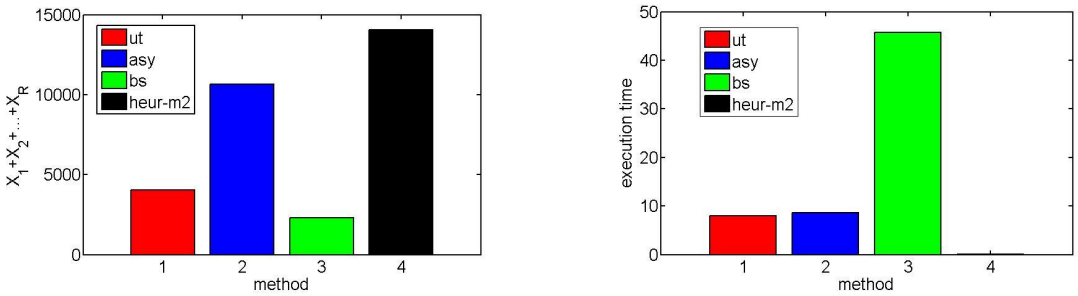
We have stressed the application using Firefox browser automation using the workload generator presented in [27]. The workload consists of a CPU-bound mix of requests. The OFBiz application is shipped in its basic form with an embedded Apache Geronimo server and an Apache embedded Derby in-memory database, which we run together inside a single VM. Each VM runs an independent copy of the application. Simultaneously, we have run the clients, one for each application, inside VMs instantiated in the same availability zone. Each experiment uses a single Firefox client coupled with a closed-loop workload generator. The time between submission of successive requests from a client is exponentially distributed with a mean of 1 second. Since there is a single client, we use a short warmup period of 2 minutes. The benchmark discards the sessions executed by the client during the warmup phase, since these can suffer high response times due to cache misses. After the warmup, the experiment continues at steady-state for 10 minutes before the server is shut down and restarted for the following experiment. Each restart takes approximately 15 minutes to regenerate the database and boot OFBiz. Thus, successive experiments inside a VM can be assumed as independent of each other. We have collected the resulting response time obtained using a single Firefox client and assumed these to be good approximations of the mean service time of a request since no other users are in execution in the system.

From the datasets collected in this way, the service rates for each workload class (i.e., request type) and deployment type have been computed and scaled by the number of virtual cores in each VM. For simplicity, we have focused on the top 15 most frequent request types, for which the number of invocations collected in the experiments was sufficient to generate robust statistics in all tests. For each class r, the job population N_r is computed as the number of clients scaled by the occurrence probability of request type r.

Using the rates determined in this way, we have parameterized a parallel processor sharing model and run ut, asy, bs, and heur-m2 on this model, assuming a maximum concurrency level of $N = 1000$ requests. The number of queues equals the number of different deployments in Table 5 and we evaluate the revenue maximization problem using the above algorithms. For ease of interpretation, we set all costs to $c_r = 1$ such that the objective function is simply the aggregate throughput of the VMs.

Experimental results are shown in Figure 2. Time are reported in seconds. The results indicate that, once more, the proposed heuristic is successful in finding a good local optimum in negligible time. In this model, the bs algorithm appears to be outperformed by our asympt method, both in terms of revenue and computational cost. Interestingly, this model has much more classes per station than the ones considered in the random experiments, so the result may suggest that bs scales worse than asympt as the number of classes grows. A reasonable explanation for this is that bs has a larger number of constraints and variables than asympt and the gap increases with the number of classes. The performance of ut is quite good given the simplicity of the formulation, but still it is not competitive with heur-m2.

Summarizing, the datasets collected from a real cloud application confirm that revenue maximization for multiclass queueing models can be analyzed very efficiently with the proposed heuristic, at negligible computational costs, also when service rates are parameterized with realistic values. Thus, while this validation is not sufficient to prove effectiveness of the methodology in revenue maximization of real cloud software systems, an interesting research direction that we leave for future work, it offers some evidence that the heuristic will not incur issues with realistic parameterizations of the service rates.



(a) Revenue comparison

(b) Computational cost

Figure 2: Revenue maximization problem parameterized with service rates from traces of a cloud application.

7 Conclusions

In this paper, we have investigated the problem of selecting routing probabilities for scheduling requests in parallel processor sharing queues. Compared to other works, we focus on multiclass workloads and closed systems. Our main result is a heuristic closed-form formula to assign such probabilities. Experimental results indicate that, on small and medium sized models, the heuristic has similar accuracy to optimization-based models, but at negligible computational costs. In large scale models, our heuristic becomes the only practical method, since optimization-based formulations do not converge to a local optimum after several minutes. Possible directions for future work include the application of the methodology to runtime system performance management and its extension to include other forms of scheduling and think times.

References

[1] E. Altman, B. Gaujal, and A. Hordijk. *Discrete-Event Control of Stochastic Networks: Multimodularity and Regularity*. Number 1829 in LNM. Springer-Verlag, 2003.

[2] J. Anselmi, P. Cremonesi. A unified framework for the bottleneck analysis of multiclass queueing networks. Perform. Eval., 67(4):218–234, 2010.

[3] J. Anselmi, B. D'Auria, and N. Walton. Closed queueing networks under congestion: Nonbottleneck independence and bottleneck convergence. *Math. Oper. Res. (to appear)*.

[4] J. Anselmi, B. Gaujal. The 'Price of Forgetting' in parallel and non-observable queues. Perform. Eval., 68(12):1291–1311, 2011.

[5] C. Badue, J. Almeida, V. Almeida, R. Baeza-Yates, B. Ribeiro-Neto, A. Ziviani, and N. Ziviani. Capacity planning for vertical search engines, Jun 2010. Preprint: http://arxiv.org/abs/1006.5059.

[6] G. Balbo and G. Serazzi. Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Perform. Eval.*, 30(3):115–152, 1997.

[7] A. Bar-Noy, R. Bhatia, J. S. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. *Math. Oper. Res.*, 27(3):518–544, 2002.

[8] F. Baskett, K. Chandy, R. Muntz, and F. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.

[9] C. H. Bell and S. Stidham. Individual versus social optimization in the allocation of customers to alternative servers. *Management Science*, 29(7):831–839, 1983.

[10] M. Bennani and D. A. Menascè. Resource allocation for autonomic data centers using analytic performance. In *Proc. of the 2nd IEEE Int. Conf. on Autonomic Computing (ICAC)*, 229–240, Jun 2005.

[11] K. P. Bennett, O. L. Mangasarian. Bilinear separation of two sets in n-space. *Comp. Optim. and App.*, 2, 1993.

[12] C. Berge. *Topological spaces*. Dover reprint, 1963.

[13] S. C. Borst. Optimal probabilistic allocation of customer types to servers. ACM SIGMETRICS '95/PERFORMANCE '95, 116–125, New York, NY, USA, 1995. ACM.

[14] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] K. M. Chandy and D. Neuse. Linearizer: A heuristic algorithm for queuing network models of computing systems. *Comm. of the ACM*, 25(2):126–134, 1982.

[16] Y. Chen, S. Iyer, X. Liu, D. S. Milojicic, and A. Sahai. Translating service level objectives to lower level policies for multi-tier services. *Cluster Computing*, 11(3):299–311, 2008.

[17] W. C. Cheng and R. R. Muntz. Optimal routing for closed queueing networks. *Perform. Eval.*, 13(1):3–15, Oct. 1991.

[18] M. B. Combé and O. J. Boxma. Optimization of static traffic allocation policies. *TCS*, 125(1):17–43, 1994.

[19] D. Cox and W. Smith. *Queues*. Methuen, London, 1961.

[20] D. M. Dakshayini and D. H. S. Guruprasad. An optimal model for priority based service scheduling policy for cloud computing environment. *Int. J. of Computer Applications*, 1(1), 2011.

[21] X. Guo, Y. Lu, M. S. Squillante. Optimal probabilistic routing in distributed parallel queues. *ACM PER*, 32(2):53–54, 2004.

[22] A. Hordijk, J. A. Loeve. Optimal static customer routing in a closed queuing network. *Statistica Neerlandica*, 54(2):148–159, 2000.

[23] F. Kelly. *Reversibility and Stochastic Networks*. Wiley, Chicester, 1979.

[24] H. Kobayashi and M. Gerla. Optimal routing in closed queuing networks. *ACM TOCS*, 1(4):294–310, Nov. 1983.

[25] J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proc. of CACSD*, 2004.

[26] M. Mazzucco, I. Mitrani, J. Palmer, M. Fisher, and P. McKee. Web service hosting and revenue maximization. In *ECOWS*, 45–54. IEEE Computer Society, 2007.

[27] J. Moschetta, G. Casale. OFBench: an Enterprise Application Benchmark for Cloud Resource Management Studies. In *MICAS* workshop, IEEE Press, 2012.

[28] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: part I – Convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.

[29] D. Menasce and V. A. F. Almeida. *Capacity Planning for Web Services.* Prentice Hall, 2002.

[30] B. Pittel. Closed exponential networks of queues with saturation: The jackson-type stationary distribution and its asymptotic analysis. *Math. Oper. Res.*, 4(4):357–378, 1979.

[31] N. Roy, A. Dubey, A. S. Gokhale, and L. W. Dowdy. A capacity planning process for performance assurance of component-based distributed systems. *Proc. ICPE*, 259–270, ACM, 2011.

[32] P. Schweitzer. Bottleneck determination in networks of queues. *Proc. ORSA/TIMS Special Interest Conf. on Appl. Probab. - Comput. Sci., The Interface, Boca Raton, FLA*, 471–485, 1981.

[33] P. J. Schweitzer. Approximate analysis of multiclass closed networks of queues. In *Proc. of the Int'l Conf. on Stoch. Control and Optim.*, 25–29, Amsterdam, 1979.

[34] J. Sethuraman and M. S. Squillante. Optimal stochastic scheduling in multiclass parallel queues. In *Proc. of SIGMETRICS*, 93–102, New York, NY, USA, 1999. ACM.

[35] J. G. Shanthikumar and D. D. Yao. Stochastic monotonicity of the queue lengths in closed queueing networks. *Oper. Res.*, 35(4):583–588, 1987.

[36] J. M. Smith. Optimal routing in closed queueing networks with state dependent queues. *INFOR: Information Systems and Operational Research*, 49(1):45–62, 2011.

[37] C. Stewart, T. Kelly, A. Zhang. Exploiting nonstationarity for performance prediction. *ACM EuroSys*, 31–44, 2007.

[38] S. K. Tripathi and C. M. Woodside. A vertex-allocation theorem for resources in queuing networks. *J. ACM*, 35(1):221–230, Jan. 1988.

[39] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An analytical model for multi-tier internet services and its applications. ACM SIGMETRICS, 291–302, New York, NY, USA, 2005. ACM.

[40] N. S. Walton. Proportional fairness and its relationship with multiclass queueing networks. *Annals of Applied Probability*, 19(6):2301–2333, 2009.

[41] W. Whitt. Open and closed models for networks of queues. *AT&T Bell Lab. Tech. J.*, 63, 9:1911–1979, 1984.

[42] L. Wu, S. K. Garg, and R. Buyya. SLA-based admission control for a software-as-a-service provider in cloud computing environments. *J. Comput. Syst. Sci*, 78(5):1280–1299, 2012.

[43] J. Zahorjan, D. L. Eager, and H. M. Sweillam. Accuracy, speed, and convergence of approximate mean value analysis. *Perform. Eval.*, 8(4):255–270, 1988.

Proof of Theorem 2

Recall that $X^*(p) \stackrel{\text{def}}{=} \lim_{k\to\infty} X(kN,p)$ is given by Proposition 2.

Lemma 1. *For all r and p, $X_r^*(p) \geq \frac{N_r}{\sum_s N_s} \min_i \mu_{ir}$.*

Proof. Since (13) is a strictly-concave optimization problem and Slater's condition holds, its KKT conditions uniquely identify the optimizer $X^*(p)$. Using that $X^*(p) > 0$, which follows because $\lim_{x\to 0^+} \log x = -\infty$, and, with respect to Lagrange multipliers L_i, $i = 1,\ldots,M$, the KKT conditions of (13) can be written as

$$\sum_{i\in H_r} L_i X_r^*(p) \tfrac{p_{ir}}{\mu_{ir}} = N_r, \qquad \forall r \tag{23a}$$

$$L_i\Big(\sum_{r:i\in H_r} \tfrac{p_{ir}}{\mu_{ir}} X_r^*(p) - 1\Big) = 0, \qquad \forall i \tag{23b}$$

$$\sum_{r:i\in H_r} \tfrac{p_{ir}}{\mu_{ir}} X_r^*(p) \leq 1, \qquad \forall i \tag{23c}$$

$$X^*(p) \geq 0, L \geq 0. \tag{23d}$$

Summing (23a) over r and using (23b), we get

$$\sum_r N_r = \sum_r \sum_{i\in H_r} L_i X_r^*(p) \tfrac{p_{ir}}{\mu_{ir}} = \sum_i L_i \sum_{r:i\in H_r} X_r^*(p) \tfrac{p_{ir}}{\mu_{ir}} = \sum_i L_i. \tag{24}$$

Now, using the former equation and that $p_{ir} \in [0,1]$, we obtain

$$X_r^*(p) \stackrel{\text{by (23a)}}{=} \frac{N_r}{\sum_{i\in H_r} L_i \frac{p_{ir}}{\mu_{ir}}} \geq \frac{N_r}{\sum_i L_i p_{ir}} \min_i \mu_{ir} \geq \frac{N_r}{\sum_i L_i} \min_i \mu_{ir} = \frac{N_r}{N} \min_i \mu_{ir}. \tag{25}$$

□

Using Lemma 1, we can rewrite (13) as

$$X^*(p) = \arg\max_{X \geq \epsilon} \sum_r N_r \log X_r \tag{26a}$$

$$\text{s.t.:} \sum_{r:i \in H_r} \tfrac{p_{ir}}{\mu_{ir}} X_r \leq 1, \qquad \forall i, \tag{26b}$$

where $\epsilon \stackrel{\text{def}}{=} (\frac{N_1}{\sum_s N_s} \min_i \mu_{i1}, \ldots, \frac{N_R}{\sum_s N_s} \min_i \mu_{iR})$. The advantage of formulation (26) is that the objective is continuous over all domain of the problem, which is used in the following lemma.

Lemma 2. *Under Assumption 1,*

$$\begin{aligned} \lim_{k\to\infty} \Pi^*(kN) = \max_{p \geq 0} \quad & \sum_r c_r X_r^*(p) \\ s.t.: \quad & \textstyle\sum_i p_{ir} = 1, \quad \forall r \\ & p_{ir} = 0, \qquad \forall i \notin H_r, r. \end{aligned} \tag{27}$$

Proof. We can exchange the lim and max operators as in the statement of this lemma if the convergence $X_r(kN,p) \to X_r^*(p)$, for all r, is uniform over the compact set $\{p \geq 0 : \sum_i p_{ir} = 1, \forall r, \text{ and } p_{ir} = 0, \forall i \notin H_r, r\}$. To establish uniform convergence, we use Dini's theorem and the monotonicity assumption. Continuity (in p) of $X_r(kN,p)$ is obvious. Continuity of $X_r^*(p)$ follows by the theorem of the Maximum [12, Chapter 6] applied to the characterization (26). The pointwise convergence of $\sum_r c_r X_r(kN,p)$ is shown in Proposition 2. Finally, $\sum_r c_r X_r(kN,p)$ is eventually monotone (in k) because $X_r(kN,p)$ is eventually monotone for each r by assumption. □

Using (27) and Proposition 2, $\Pi^* \stackrel{\text{def}}{=} \lim_{k\to\infty} \Pi^*(kN)$ exists by Weierstrass theorem because the $X_r^*(p)$'s are continuous (and thus their sum) by the theorem of the Maximum and the maximization is over a compact set. Substituting in (27) the KKT conditions of problem (13) written with respect to Lagrange multipliers L_i, $i = 1, \ldots, M$, that are given in (23), we obtain

$$\begin{aligned} \Pi^* = \max_{p\geq 0, X \geq 0, L \geq 0} \quad & \textstyle\sum_r c_r X_r & \\ \text{s.t.:} \quad & \textstyle\sum_i p_{ir} = 1, & \forall r \\ & p_{ir} = 0, & \forall i \notin H_r, r \\ & \textstyle\sum_{i \in H_r} L_i X_r \frac{p_{ir}}{\mu_{ir}} = N_r, & \forall r \\ & \textstyle L_i(\sum_{r:i\in H_r} \frac{p_{ir}}{\mu_{ir}} X_r - 1) = 0, & \forall i \\ & \textstyle\sum_{r:i\in H_r} \frac{p_{ir}}{\mu_{ir}} X_r \leq 1, & \forall i \end{aligned} \tag{28}$$

The following lemma simplifies the structure of the optimization in (28).

Lemma 3. *Let (p, X, L) be an optimal solution of problem (28). Then, $\sum_{r:i\in H_r} \frac{p_{ir}}{\mu_{ir}} X_r = 1$, $\forall i$.*

Proof. The idea for this proof is to recognize that $\sum_r X_r^*(p)$ can be interpreted as the throughput of a *single-class* network that we construct in the following. For single-class networks, the throughput is known to satisfy a simple relation that we use to conclude this proof.

Associated with the multiclass queueing network under investigation with R classes, we construct another queueing network with only one class. This new network is formed by some (not defined) service rates $\tilde{\mu}_i$ and some (not defined) routing probabilities $\tilde{p}_i$ such that their ratios $\rho_i \stackrel{\text{def}}{=} \tilde{p}_i/\tilde{\mu}_i$ are given by

$$\rho_i = \rho_i(p) = \frac{\sum_r X_r^*(p) \frac{p_{ir}}{\mu_{ir}}}{\sum_r X_r^*(p)}, \ \forall i. \tag{29}$$

Note that the $\rho_i(p)$'s are well-defined because of the strict concavity of (13). For a closed, single-class network, it is well-known that the asymptotic throughput is $1/\max_i \rho_i$ (this also follows by Proposition 2, assuming $R = 1$). In particular, the asymptotic throughput of the single-class network that we have constructed is

$$\min_i \frac{1}{\rho_i(p)} = \min_i \frac{\sum_r X_r^*(p)}{\sum_r X_r^*(p) \frac{p_{ir}}{\mu_{ir}}} = \sum_r X_r^*(p), \tag{30}$$

where the last equality follows because at least one queue of the multiclass network saturates (in the limit). This is evident because the maximizer of (13) is on the boundary of the feasibility region for any p. This means that the asymptotic throughput of the single-class network is equal to the overall asymptotic throughput of the multiclass network, for any p. Assuming $c_i = 1$ for all i, this means that we have

$$\begin{array}{rl} \Pi^* & = \max\limits_{p\geq 0} \sum_r X_r^*(p) \text{ s.t.: } \sum_i p_{ir} = 1,\, \forall r,\, p_{ir} = 0, \forall i \notin H_r, r \\ & = \max\limits_{p\geq 0} \min_i \frac{1}{\rho_i(p)} \text{ s.t.: } \sum_i p_{ir} = 1,\, \forall r,\, p_{ir} = 0, \forall i \notin H_r, r. \end{array} \tag{31}$$

Lemma 4. *Let $c_i = 1$ for all i, and let p be an optimal solution of (31). Then, $\rho_i(p) \sum_r X_r^*(p) = 1$, for all i.*

Proof. By contradiction, assume that $\rho_i(p) < \rho_j(p)$ for $i \neq j$. Then, $\min_i \frac{1}{\rho_i(p)}$ is not maximized as it is possible to make $\rho_j(p)$ smaller by decreasing $(p_{j1}, \ldots, p_{jR})$. Thus, it must be $\rho_i(p) = \rho_j(p)$. Formula (30) proves the lemma. □

Given that the p maximizes (31) if and only if it maximizes (28), Lemma 4 implies that in an optimal solution (X, L, p) of (28) we have $\sum_{r:i\in H_r} X_r \frac{p_{ir}}{\mu_{ir}} = 1$.

Now, the case where the c_r's can be different is easily dealt with by the change of variable $x_r = c_r X_r$, for all r, in formulation (28). We obtain

$$\begin{array}{rll} \Pi^* = \max\limits_{p,x,L\geq 0} & \sum_r x_r & \\ \text{s.t.:} & \sum_i p_{ir} = 1, & \forall r \\ & p_{ir} = 0, & \forall i \notin H_r, r \\ & \sum_{i\in H_r} L_i \frac{x_r}{c_r} \frac{p_{ir}}{\mu_{ir}} = N_r, & \forall r \\ & L_i(\sum_{r:i\in H_r} \frac{p_{ir}}{\mu_{ir}} \frac{x_r}{c_r} - 1) = 0, & \forall i \\ & \sum_{r:i\in H_r} \frac{p_{ir}}{\mu_{ir}} \frac{x_r}{c_r} \leq 1, & \forall i \end{array} \tag{32}$$

which is interpreted as the optimization problem (28) but referred to as a multiclass network with R classes where, w.r.t. the original network, all the class-r service rates are sped up by a factor of c_r. Therefore, we can reuse the argument above based on a single-class network to conclude that Lemma 4 also holds when the c_r can be different. □

Using Lemma 3, we rewrite (28) as

$$\begin{array}{rll} \Pi^* = \max\limits_{p,X,L\geq 0} & \sum_r c_r X_r & \\ \text{s.t.:} & \sum_i p_{ir} = 1, & \forall r \\ & p_{ir} = 0, & \forall i \notin H_r, r \\ & \sum_{i\in H_r} L_i \frac{p_{ir}}{\mu_{ir}} X_r = N_r, & \forall i \\ & \sum_{r:i\in H_r} \frac{p_{ir}}{\mu_{ir}} X_r = 1, & \forall i. \end{array} \tag{33}$$

Introducing the variables X_{ir} and making the substitutions $X_{ir} = p_{ir} X_r$ for all i and r, we obtain that (33) is equivalent to (18).

Now, to prove (19), one can easily check that the KKT conditions of optimization problem (13) when the routing probability matrix is p^* (defined in the statement of the theorem) are equivalent to those of program (18).