Probability in the Engineering and Informational Sciences, **15**, 2001, 165–178. Printed in the U.S.A.

ON THE COMPARISON OF QUEUEING SYSTEMS WITH THEIR FLUID LIMITS

EITAN ALTMAN
INRIA
06902 Sophia Antipolis, France
E-mail: altman@sophia.inria.fr

TANIA JIMÉNEZ
INRIA
06902 Sophia Antipolis, France
and
CeSiMo
Universidad de Los Andes
Mérida, Venezuela
E-mail: tania@ula.ve

GER KOOLE
Vrije Universiteit
1081 HV Amsterdam, The Netherlands
E-mail: koole@cs.vu.nl

In this article, we study a method to compare queueing systems and their fluid limits. For a certain class of queueing systems, it is shown that the expected workload (and certain functions of the workload) is higher in the queueing system than in the fluid approximation. This class is characterized by convexity of the value function in the state component(s) where external arrivals occur. The main example that we consider is a tandem of multiserver queues with general service times and Markov-modulated arrivals. The analysis is based on dynamic programming and the use of phase-type distributions. Numerical examples to illustrate the results are also given.

1. INTRODUCTION

Queueing theory is mainly concerned with calculating the influence of randomness in certain processes as a function of certain stationary input processes that model arrivals and departures. Indeed, the variance in, for example, service times has a great influence on waiting times, as is nicely illustrated for the $M/G/1$ queue in the celebrated Pollaczek–Khitchine formula.

© 2001 Cambridge University Press 0269-9648/01 $12.50

However, in certain situations, we do not wish to make the stationarity assumption, but to study the influence of changes in, for example, the arrival rate. In such cases, exact queueing results, taking into account nonstationarity and variances, are extremely hard to obtain. An "engineering approach" [6, p. 56] to this problem is to focus on the nonstationarity and assume continuous deterministic arrival and departure flows. This leads to the well-known fluid approximation for queues. This approximation is usually easier to compute than the original queueing system and might be of interest on its own.

However, often we ask ourselves how close the fluid approximation is to the original model. In this article, we try to answer part of this question by showing that for certain systems and performance measures, the fluid limit performs better than the original queueing system. Our main example is a tandem of multiserver queues and performance measures such as total (weighted) workload. The method applies equally well to other classes of models. Service times are general but fixed for each customer. There is an environment modulating the arrival rates and the server speeds.

In Section 2, our main result is formulated in detail for the tandem system. This result is obtained in Sections 3–5 as follows. First, we study exponential queueing models. This queueing model, introduced in Section 3, consists of a tandem of multiserver queues. Both the arrival rate and the service rates are governed by an underlying Markov process. Arrivals can occur in batches.

This model is defined by its dynamic programming recursion. In Section 4, dynamic programming is used to compare two tandem queueing models that differ in the workload: One system has batch size 1, the other possibly higher batch sizes but less frequent arrivals. We take the arrival rates and the number of exponential phases per customer such that the average workload to the systems is equal. We give conditions on the performance measures for which the system with batch arrivals behaves worse. In Section 5, the fluid limit is studied. By increasing service rates, arrival rates, and batch sizes in the correct way, the queueing model without batches converges to a fluid model, and the model with batches converges to a workload model with arbitrary service times.

In Section 6, we show how the method can be applied to other queueing systems. Examples are a single queue with an admission control and a two-queue routing model.

The comparisons are illustrated with the analysis of a few example systems in Section 7. One is a single-queue model with a nonstationary arrival rate, motivated by Newell [13]. Another is the comparison of standard homogeneous $M/M/1$ and $M/D/1$ queues and their fluid approximations with a large initial workload. The last numerical example consists of a tandem system with two queues with a large amount of initial workload in the first one.

2. PERFORMANCE MEASURES AND MAIN RESULT

In this section, we state the main result for the tandem system. We give conditions on the performance measures under which the fluid limit has a performance lower than

the original queueing system. These conditions are formulated in terms of sets of functions. We first discuss these sets. For the moment, all we need to know about our system is that it consists of m queues and an environment.

All performance measures (or costs, in dynamic programming terminology) are functions of the whole state space, which is given by $\mathbb{N}_0 \times \mathbb{R}_+^m$. The 0th (integer) state component represents the environment and the ith state component ($1 \le i \le m$) represents the workload in the ith queue. Each performance measure is an element of the set $\mathcal{F} = \{ f \mid f: \mathbb{N}_0 \times \mathbb{R}_+^m \to \mathbb{R}\}$.

DEFINITION 2.1: *A function* $f: \mathbb{R}_+^m \to \mathbb{R}$ *is called* directionally convex *if for all* $x, y, z \in \mathbb{R}_+^m$

$$f(x+y) + f(x+z) \le f(x) + f(x+y+z).$$

Directional convexity was introduced in [12], where it is formulated in a slightly different (but equivalent) way.

To keep terminology simple, we call a function $f \in \mathcal{F}$ also directionally convex (written as $f \in \mathcal{F}(\mathrm{dc})$) if f is directionally convex for each environment state. This assures that if $f \in \mathcal{F}(\mathrm{dc})$, then there is no condition on the way f depends on the environment.

Many performance measures are directionally convex. Note that each performance measure that is additive and convex in the components is also directionally convex.

DEFINITION 2.2: *A function* $f: \mathbb{R}_+^m \to \mathbb{R}$ *is called* downstream decreasing *if for all* $1 \le i < j \le m$,

$$f(x + e_j) \le f(x + e_i)$$

and if

$$f(x) \le f(x + e_m).$$

Again, we call a function $f \in \mathcal{F}$ also downstream decreasing (written as $f \in \mathcal{F}(\mathrm{dd})$) if f is downstream decreasing for a fixed environment state. It is easy to show that every downstream decreasing function is increasing (terms such as *increasing* and *decreasing* are used in the nonstrict sense) in each of the variables. We write $\mathcal{F}(\mathrm{dc,dd}) = \mathcal{F}(\mathrm{dc}) \cap \mathcal{F}(\mathrm{dd})$.

Costs that are downstream decreasing imply that it is favorable to have jobs further down the line. This is the case in many systems. Some examples where it is not the case is in systems where holding costs increase down the line, as in some production systems, and in systems with finite buffers, modeled by additional costs if queue lengths exceed buffer levels.

Remark 2.3: The class of functions $\mathcal{F}(\mathrm{dd})$ establishes also an ordering between the states: $\mathcal{F}(\mathrm{dd})$ consists of all functions that are increasing in the *partial sum ordering* (see [3, Lemma 2.7] and [7, Thm. C.2]).

We describe the model in words. We consider a tandem system consisting of $m \in \mathbb{N}$ multiserver queues and arrivals at one or more queues. Customers have general service times (with average 1) that remain equal at the different stages of the tandem system. The speeds at which the servers at different queues work can differ. There is an independent Markovian environment governing arrival rates and server speeds. The fluid model consists also of m queues and the same Markovian environment. The arrival rates are now the speeds at which fluid enters the system.

Note that the condition that the average service time is 1 does not constrain the model: By rescaling, we can get any average. The constraint is that the queueing and the fluid model have the same expected offered workload.

Our main result now follows. It will be proven in the next three sections. For an initial workload w, let W_t be the workload vector of the queueing system at time t. Similarly, let Z_t be the workload of the fluid model at time t, also for the initial state w. With U_t, we indicate the state of the environment at t, and $U_0 = u$.

THEOREM 2.4: *If* $C \in \mathcal{F}(\mathrm{dc},\mathrm{dd})$ *and if* $\inf_u C(u,0) > -\infty$, *then*

$$\mathbb{E}_{u,w} C(U_t, Z_t) \leq \mathbb{E}_{u,w} C(U_t, W_t).$$

3. THE EXPONENTIAL MODEL

To derive Theorem 2.4, we first study exponential queueing models with possibly batch arrivals. In this section, we introduce these systems in detail. They consist of $m \in \mathbb{N}$ queues and an environment. The environment is modeled by an arbitrary continuous-time Markov process on $\mathbb{N}_0$, with transition rates $\gamma(u,v)$, with $\sum_{v=0}^{\infty} \gamma(u,v)$ uniformly bounded. This Markov process governs the transitions at the queueing system. At each of the m queues, arrivals can occur, at a rate $\lambda^j(u)$ at queue j, with u the current state of the environment. We assume that $\sup_u \lambda^j(u) < \infty$ for each j. Queue j, $1 \leq j \leq m$, has S_j servers that serve at a rate $\mu^j(u)$ each. Again, $\sup_u \mu^j(u) < \infty$.This specifies the model without batch arrivals. The model with batch arrivals will be introduced later in this section.

Let $X_t \in \mathbb{N}_0^m$ denote the number of jobs in the queues at t. Next, we develop a dynamic programming formulation for $\mathbb{E}_{u,x} C(U_t, X_t)$, the expected costs at t for the initial state of the system $(u,x) \in \mathbb{N}_0^{m+1}$. Note that $C : \mathbb{N}_0^{m+1} \to \mathbb{R}$ in this section and in the next. Because all transition rates are uniformly bounded, we can rewrite them, for some $\alpha > 0$, as $\gamma(u,v) = \alpha\bar{\gamma}(u,v)$, $\lambda^j(u) = \alpha\bar{\lambda}^j(u)$, and $\mu^j(u) = \alpha\bar{\mu}^j(u)$, with $\sum_{v=0}^{\infty} \bar{\gamma}(u,v) + \sum_{j=1}^{m} (\bar{\lambda}^j(u) + S_j \bar{\mu}^j(u)) \leq 1$. By adding a dummy transition to the environment [i.e., giving $\bar{\gamma}(u,u)$ a positive value], we can assure that $\sum_{v=0}^{\infty} \bar{\gamma}(u,v) + \sum_{j=1}^{m} [\bar{\lambda}^j(u) + S_j \bar{\mu}^j(u)] = 1$ for every u.

Now, we can see the transitions of the system as follows: According to a Poisson process, at rate α, jumps are generated, and at the jump times, the system changes state according to a discrete-time Markov chain with transition probabilities $\bar{\gamma}(u,v)$, $\bar{\lambda}^j(u)$, and $\bar{\mu}^j(u)$, where one or more "dummy departures" occur at queue j if the state is such that there are less than S_j jobs in queue j. This representation of the system

will help us find a dynamic programming formulation for $\mathbb{E}_{u,x}C(U_t, X_t)$, the expected costs at t.

First, however, we introduce a condition that assures that $\mathbb{E}_{u,x}C(U_t, X_t)$ is integrable for all $C \in \mathcal{F}(\mathrm{dd})$.

LEMMA 3.1: *For all* $C \in \mathcal{F}(\mathrm{dd})$ *that satisfy* $\inf_{u\in\mathbb{N}_0} C(u,0) > -\infty$, $\mathbb{E}_{u,x}C(U_t, X_t)$ *exists and is finite or* ∞.

PROOF: Every $C \in \mathcal{F}(\mathrm{dd})$ is increasing in all its components; therefore, $C(u,0) \le C(u,x)$ for all x and u. Thus, $\inf_u C(u,0) = \inf_{u,x} C(u,x)$, and $\mathbb{E}_{u,x}C(U_t, X_t) \ge \inf_{u\in\mathbb{N}_0} C(u,0)$. ■

By conditioning on the number of jumps of the Poisson process in $[0, t]$, we can write $\mathbb{E}_{u,x}C(U_t, X_t)$ as

$$\mathbb{E}_{u,x}C(U_t, X_t) = \sum_{n=0}^{\infty} \frac{(\alpha t)^n}{n!} e^{-\alpha t} V_n(u,x), \tag{1}$$

with $V_n(u,x)$ the expected costs after n steps of the discrete-time model. For $V_n(u,x)$, we will now give a dynamic programming recursion. (The same method is also used, for different applications, in [7, Sect. 5.2] and [10, Sect. 2.3].)

For $n = 0$, we have to take $V_0(u,x) = C(u,x)$, with C the performance measure or costs. For $n > 0$, the value function V_n can be recursively defined by

$$\begin{aligned} &V_{n+1}(u,x) \\ &\quad = T_{\mathrm{env}}(V_n, T_{A(1)}(V_n), \ldots, T_{A(m)}(V_n), T_{\mathrm{TS}(1)}(V_n), \ldots, T_{\mathrm{TS}(m-1)}(V_n), T_{D(m)}(V_n))(u,x), \end{aligned}$$

with

$$T_{A(i)} f(u,x) = f(u, x + e_i),$$

$$T_{D(i)} f(u,x) = \begin{cases} x_i f(u, x - e_i) + (S_i - x_i) f(u,x) & \text{if } x_i < S_i \\ S_i f(u, x - e_i) & \text{otherwise,} \end{cases}$$

$$T_{\mathrm{TS}(i)} f(u,x) = \begin{cases} x_i f(u, x - e_i + e_{i+1}) + (S_i - x_i) f(u,x) & \text{if } x_i < S_i \\ S_i f(u, x - e_i + e_{i+1}) & \text{otherwise,} \end{cases}$$

$$\begin{aligned} T_{\mathrm{env}}(f_0, \ldots, f_{2m})(u,x) &= \sum_{v\in\mathbb{N}_0} \bar{\gamma}(u,v) f_0(v,x) \\ &\quad + \sum_{j=1}^{m} [\bar{\lambda}^j(u) f_j(u,x) + \bar{\mu}^j(u) f_{m+j}(u,x)]. \end{aligned}$$

Let us interpret this definition of V_{n+1}. All T's stand for possible events in the system. T_{env}, the environment operator, specifies the type of event that occurs at this jump. If it is a change of state of the environment, then the state changes from (u,x) to (v,x) and there are n more jumps to go, thus expected costs $V_n(v,x)$. This is reflected by the fact that by inserting the definition of T_{env} in V_{n+1}, we find $f_0 = V_n$. If, on the other hand, an arrival occurs, say at queue j, then the next event $T_{A(j)}$

occurs, which is applied to V_n. The same holds for $T_{\mathrm{TS}(i)}$, the event operator corresponding to S servers moving customers from queue i to queue $i+1$ (TS stands for "tandem server"), and $T_{D(m)}$, the event operator modeling the departures from queue m. Note that the coefficients sum to 1 because the coefficients in the departure operators sum to the number of servers.

Splitting up the dynamic programming equation into smaller event operators as we just did simplifies the proofs in the next section. For full details of this method, we refer to [8]; here, we use it in a simplified form, adapted to the comparison of different systems.

We finish this section by introducing a second system, with value function V_n^B. It differs from the system with value function V_n in the arrivals at queue i. Under V_n^B, arrivals occur less frequently, but in batches. To be more specific, $T_{A(i)}$ of V_n is replaced by

$$T_{BA(i)} f(u,x) = \sum_{k=0}^{\infty} p_k^i f(u, x + ke_i).$$

Here, we take the $p_k^i \geq 0$ such that $\sum_{k=0}^{\infty} p_k^i = 1$. Define $\bar{p}^i = \sum_{k=0}^{\infty} k p_k^i$.

In the next section, we compare V_n and V_n^B. However, let us try to understand why $T_{BA(i)}$ is an interesting operator to consider. It is known [2,15] that any probability distribution can be approximated arbitrarily close by a mixture of Erlang or gamma distributions, all with the same parameter. With $T_{BA(i)}$, we can model this approximation. When we increase the accuracy of the approximation, the batch sizes increase, but the service time per exponential unit decreases. In the model without batch arrivals, we keep the expected number of exponential units (almost) equal, meaning an increasing arrival rate. Of course, the service time per exponential unit decreases and, thus, the total expected offered workload remains the same and converges to a fluid process, as we will see in Section 5. We show in the next section that if $C \in \mathcal{F}(\mathrm{dc},\mathrm{dd})$ and $\bar{p}^i \geq 1$ for all i, then $V_n(u,x) \leq V_n^B(u,x)$ for all n, u, and x. From this, the comparison at t follows.

4. COMPARISON

Let X_t^B denote the state of the system with batch arrivals at time t. The main result of this section is the following theorem.

THEOREM 4.1: *If* $C \in \mathcal{F}(\mathrm{dc},\mathrm{dd})$, $\bar{p}^i \geq 1$ *for all* i, *and* $\inf_{u \in \mathbb{N}_0} C(u,0) > -\infty$, *then*

$$\mathbb{E}_{u,x} C(U_t, X_t) \leq \mathbb{E}_{u,x} C(U_t, X_t^B).$$

PROOF: In Theorem 4.5, we will show that $V_n(u,x) \leq V_n^B(u,x)$ if $C \in \mathcal{F}(\mathrm{dc},\mathrm{dd})$ for all n. Then, the result follows immediately by using (1) and observing that this equation also holds for $\mathbb{E}_{u,x} C(U_t, X_t^B)$. ■

In the proof of Theorem 4.5, we make use of certain properties of directionally convex functions and of the fact that V_n itself is directionally convex. The latter is shown for a more general model in [9, Cor. 4.6].

We will use the following useful characterization of directionally convexity on $\mathbb{N}_0^m$ (e.g., see [9]). We call a function f *componentwise convex* in component i if

$$2f(x + e_i) \leq f(x) + f(x + 2e_i)$$

for all x, and *supermodular* in i and j if

$$f(x + e_i) + f(x + e_j) \leq f(x) + f(x + e_i + e_j)$$

for all x. Lemma 3.2 of [9] states that a function is directionally convex if and only if it is componentwise convex and supermodular in all components i with $1 \leq i, j \leq m$. (Note that componentwise convexity is, in fact, supermodularity with $i = j$.)

Furthermore, in Lemma 3.8 of [9], it is shown that every downstream decreasing function is also increasing in each component [i.e., $f(x) \leq f(x + e_i)$ for all x and i]. The following lemma is a special case of Corollary 4.6 of [9].

LEMMA 4.2: *If* $C \in \mathcal{F}$(dc,dd), *then* $V_n \in \mathcal{F}$(dc,dd).

The difference between the last part of Corollary 4.6 in [9] and the current theorem is that the environment in Corollary 4.6 is more general; also, a Markov arrival process can be modeled with it. This is not useful in the context of fluid queues.

Now, we show certain properties of componentwise convexity, as defined already in Section 2. We denote componentwise convexity in i by $CC(i)$.

For a function $f: \mathbb{N}_0^m \to \mathbb{R}$, we define $\bar{f}^i: \mathbb{R}_+^m \to \mathbb{R}$ to be the function which coincides with f on $\mathbb{N}_0^m$ and is the linear interpolation of f along the direction i. If f is $CC(i)$, then the derivatives of $\bar{f}^i$ in direction i are clearly monotone increasing, so that

$$f \text{ is } CC(i) \Rightarrow \bar{f}^i \text{ is convex in direction } i. \tag{2}$$

This implies the following lemma.

LEMMA 4.3: *The function* f *is* $CC(i)$ *if and only if*

$$f(x + \alpha k e_i) \leq (1 - \alpha) f(x) + \alpha f(x + k e_i),$$

for all $x \in \mathbb{N}^m$, *and* $k \in \mathbb{N}$ *and* $\alpha \in [0,1]$ *such that* $\alpha k \in \mathbb{N}$.

PROOF: The "if" part follows directly by taking $k = 2$ and $\alpha = \frac{1}{2}$. The "only if" part follows from (2). ■

The inequality proven in the next lemma will prove to be crucial in the proof of Theorem 4.5.

LEMMA 4.4: *If* f *is* $CC(i)$ *and increasing in* i, $\sum_{k=0}^{\infty} p_k = 1$, *and* $\sum_{k=0}^{\infty} k p_k \geq 1$, *then*

$$f(x + e_i) \leq \sum_{k=1}^{\infty} p_k f(x + k e_i).$$

PROOF: Since $\sum_{k=0}^{\infty} kp_k \geq 1$, then, by Jensen's inequality, we have

$$f(x+e_i) = \bar{f}^i(x+e_i) \leq \sum_{k=1}^{\infty} p_k \bar{f}^i(x+ke_i) = \sum_{k=1}^{\infty} p_k f(x+ke_i). \qquad \blacksquare$$

THEOREM 4.5: *If* $C \in \mathcal{F}(\text{dc,dd})$ *and* $\bar{p}^i \geq 1$, *then* $V_n(u,x) \leq V_n^B(u,x)$ *for all* $n \in \mathbb{N}_0$.

PROOF: We prove the theorem by induction to n. For $n = 0$, we have $V_0(u,x) = V_0^B(u,x) = C(u,x)$. Assume that it is proven up to $n \geq 0$. Consider V_{n+1}. From $V_n(u,x) \leq V_n^B(u,x)$, it follows directly that $T_{A(i)}V_n(u,x) \leq T_{A(i)}V_n^B(u,x)$, $T_{\text{TS}(i)}V_n(u,x) \leq T_{\text{TS}(i)}V_n^B(u,x)$, and $T_{D(m)}V_n(u,x) \leq T_{D(m)}V_n^B(u,x)$. The crucial step is showing $T_{A(i)}V_n(u,x) \leq T_{BA(i)}V_n^B(u,x)$. As $C \in \mathcal{F}(\text{dc,dd})$, it follows from Lemmas 4.2 and 4.4 that

$$V_n(u,x+e_i) \leq \sum_{k=0}^{\infty} p_k V_n(u,x+ke_i).$$

By induction, $V_n(u,x) \leq V_n^B(u,x)$ for all (u,x), and thus

$$V_n(u,x+e_i) \leq \sum_{k=0}^{\infty} p_k V_n^B(u,x+ke_i)$$

holds. This is exactly $T_{A(i)}V_n(u,x) \leq T_{BA(i)}V_n^B(u,x)$. As T_{env} simply consists of a convex combination of terms for which the inequality holds, $V_{n+1}(u,x) \leq V_{n+1}^B(u,x)$ easily follows. $\blacksquare$

We worked out the comparison for a single operator. However, in the definition of the queueing system, we allowed for multiple arrival operators. By repeating the procedure, we can include other arrival operators in the comparison. As this is straightforward, we will not present detailed results.

We saw in the proof of Theorem 4.5 that we need that V_n is *CC* in the components that have batch and nonbatch arrivals. There are other systems than the tandem system studied here that have *CC* value functions. We will discuss these models and the results that can be obtained for them in Section 6.

5. FLUID LIMITS

There are different ways to construct and interpret fluid limits. We prefer to see them as a means to study transient behavior, such as peak hours. This is the classical interpretation (see, e.g., [6,13]). The nonstationarity can be modeled effectively with the environment. For this reason, a construction involving scaling time (such as in [4]) is not appropriate, as it would mean scaling the environment as well. Instead, we see the fluid limit as a simultaneous increase of the arrival rate and a decrease of the size of the workload per job. This corresponds to the average over an infinite number of realizations of the arrival process, for each realization of the environment separately.

We construct a series of comparisons between two queueing systems that is such that one of the better system converges to the fluid limit and the worse system to the workload vector. The comparison holds at each stage by Theorem 4.1.

We assume that arrivals occur only at queue 1. Adding arrivals to other queues is straightforward but notationally cumbersome.

Before going into the details of the comparison, we define what we call a fluid process. We say that the offered workload O_t is a fluid process generated by the environment U_t if there is a rate function $\lambda : \mathbb{N}_0 \to \mathbb{R}_+$ such that $O_t = \int_0^t \lambda(U_s)\, ds$. Thus, a fluid is characterized by the absence of jumps and, for a given realization of the environment process, a deterministic growth.

Now, add an index k to the environment, giving a series $\{U_t^k\}_{k=1}^{\infty}$, where U_t^k is equal to U_t except that all $\lambda^1(u)$ and $\mu^j(u)$ are multiplied by a factor k. Consider the queueing system without batch arrivals. We multiplied the server speeds by k. An alternative of looking at this system is assuming that the server speeds remained unchanged, but that each exponential task has a service time divided by k (i.e., with rate k). Thus, the offered workload generated by U_t^k up to t consists of a number of exponential phases with rate k, where this number is Poisson distributed with mean $\int_0^t k\lambda^1(U_s)\, ds$. Then, from standard properties of the Erlang distribution, the offered workload converges, as $k \to \infty$, to a fluid with rate $\lambda^1(U_t)$.

Next, we consider the system with batch arrivals. We want to have the same batch arrival process, independent of the environment. This is achieved by taking $p_0^1 = 1 - 1/k$ for the kth approximation. Thus, on average, $T_{BA(1)}$ selects 1 out of k points from the process with rate $k\lambda^1(U_t)$. This results in an arrival process with rate $\lambda^1(U_t)$. The service time distribution, with distribution function G, is approximated by a series of mixtures of Erlang distributions, with distribution function G^k of the kth approximate defined by

$$G^k(x) = \sum_{n=1}^{\infty} \beta_n^k E_n^k(x),$$

where $\beta_n^k = G(n/k) - G((n-1)/k)$ and $E_n^k(x)$ is the distribution function of the Erlang distribution with rate k and n phases. The decrease in service time of the exponential phases is already incorporated in the server speeds. The operator $T_{BA(1)}$ generates the right number of phases if we take $p_n^1 = \beta_n^k/k$ for all $n > 0$. It is important to note that $\bar{p}^1 = \sum_{n=0}^{\infty} np_n^1 = \sum_{n=0}^{\infty} n/k(G(n/k) - G((n-1)/k)) \geq \int_0^{\infty} dG(x)$. As the average service time equals 1, we find $\bar{p}^1 \geq 1$. In other words, for suitable C, we are allowed to apply Theorem 4.1 for every U_t^k and $T_{BA(1)}$ defined earlier. This means that the comparison holds for the limit as well. We have already shown that the queueing model without batch arrivals converges to the fluid model. Thus, it remains to show that the batch arrival model converges to the original queueing model. As the workload at t is a continuous function of the input, it suffices to show that G^k converges to G. This result, however, is well known and can be found in [15] (see also [7, App. A]). It looks as if we have finished proving Theorem 2.4; there is one complication left however.

In X_t^B, the tasks of which a batch consists are treated separately, although they should be treated as a whole. In the current systems, one task of a batch can be processed at one stage while another task is already processed at the next stage. The same holds for multiserver queues: Tasks of the same customer can be processed in parallel. Thus, the correct system is yet another that delays certain tasks until all tasks of a customer have finished processing.

The solution is as follows. We introduce yet another system that incorporates the delays. To show that the real system is even worse than the one we already studied with T_{BA}, we have to show that it is better to have tasks further down the line. For performance measures in $\mathcal{F}(\mathrm{dd})$, it can easily be shown using a coupling argument that the delayed system has a worse performance. This completes the proof of Theorem 2.4.

6. EXTENSIONS TO OTHER MODELS

In [8], a unifying approach and an overview is given of monotonicity results for one- and two-dimensional (queueing) systems. For models with more than two dimensions, there are few results in addition to the ones already cited, with the exception of the cycle of queues of [16] and the fork-join queue of [1]. Many of the monotonicity results for these models are obtained by proving properties that are stronger than *CC*, thereby allowing one to obtain similar results as for the tandem system of the previous sections. We give a few examples.

6.1. One-Dimensional Models

The main class of one-dimensional models consists of models with operators that propagate convexity and increasingness (CI) in the single-state component. It contains operators that model admission control, multiple servers, a single server with a controllable rate, and so forth. As long as the arrival process is modeled by the $T_{A(1)}$ operator (i.e., uncontrolled, no finite buffers, etc.), we find by a completely analogous argument as earlier that the performance is better in the fluid approximation than in the original queueing system, assuming that the costs are CI. Here, we should note that $T_{BA(1)}$ propagates the CI property. This is easily seen, as $T_{BA(1)}$ can be seen as a random number of convolutions of $T_{A(1)}$ operators.

The same does not hold for the well-known admission control operator. The batch arrival operator T_{BAC} defined by

$$T_{\mathrm{BAC}} f(x) = \min\left\{c + f(x), \sum_{k=0}^{\infty} p_k f(x+k)\right\},$$

does not propagate the CI property. For the same reason, we have no results for models with other arrival operators such as the one modeling a finite buffer.

6.2. Two-Dimensional Models

What holds for the one-dimensional models holds also for the two-dimensional models: We can only deal with models that have $T_{A(i)}$-type arrival operators. A model

from the literature that has this property is the model of [11]. It consists of a single-server queue with an additional server that can be used if necessary. Using event-based dynamic programming techniques, it can be shown that the optimal policy is of a threshold type and that the value function is *CC* (see [8]). Thus, the comparison result holds also in this case.

6.3. Higher Dimensional Models

The prime example of monotonicity of a multidimensional queueing model (in addition to the model of [9] utilized in this article) is the tandem of queues with controlled service rates of [16]. The set of functions that they use (the multimodular functions) is contained in $\mathcal{F}$(dc,dd); therefore, the comparison result holds as well for the model of [16]. They allow only for uncontrolled arrivals at the first queue of the tandem system.

It is shown in [1] (based on ideas from [5]) that the value function of the fork-join queue is also multimodular. However, arrivals generate, here, an arrival in each queue: $T_{\mathrm{FJA}} f(x) = f(x + e)$, with $e = (1,\ldots,1)$. Thus, to show $V_n(x) \leq V_n^B(x)$, we need that the value function is convex in the e direction [i.e., that the value function satisfies $2f(x+e) \leq f(x) + f(x+2e)$]. This follows from the multimodularity of the value function. Therefore, the comparison result holds for the fork-join queue.

7. NUMERICAL EXAMPLES

In Section 1, we stated that in queueing, we can distinguish two types of variation: changes in the system parameters such as arrival and service rates, and the random changes in interarrival and service times. The first type of variation is modeled by the environment, and, therefore, it already shows up in the fluid approximation. The queueing system also captures the second type of variation.

In this section, we compare queueing systems and their fluid approximations for several configurations. First, note that a standard stable $M/G/1$ queue has fluid approximation 0 for an initially empty system. If there is some initial workload, then the system will reach and stay at 0 after some point in time. This shows that it is only interesting to consider fluid approximations if there is a large initial workload or if there are time intervals at which the offered workload is higher than the service capacity in one or more of the queues. In the configurations that we consider, one of the two situations will always be the case. In all cases, we take a deterministic environment. Of course, this cannot be modeled immediately by the Markovian environment that we studied in previous sections. However, using arguments comparable to those in [2], we see that any deterministic environment can be approximated arbitrarily close by a Markovian environment.

The first example is in the same spirit as that of [13, Fig. 2.2] and [6, Fig. 2.7]. It consists of a single $M/M/1$ queue with a constant service rate of 0.4, and arrival rate 0.8 up to time 100 and arrival rate 0.2 after 100. The initial workload is equal to 0. For this system, it is trivial to calculate the fluid approximation. Using simulation,

we calculated the expected workload at each time up to 600. The results can be found in Figure 1. The queueing system is the average over 600 traces. We see that the average workload in the queueing system is always higher than the workload in the fluid system, as predicted by Theorem 2.4. We also see that the behavior is similar for large queue lengths. We also see that the fluid is equal to 0 from 300 on, and the expected workload in the $M/M/1$ system converges to 2.5, the stationary workload of the $M/M/1$ queue with rates 0.2 and 0.4. The fluid approximation is independent of the service time distribution; it depends only on the expectation of the service time. However, the workload in the $M/G/1$ queue depends strongly on the form of the service time distribution, as we know from the Pollaczek–Khitchine formula. It is interesting to consider, as well, the $M/D/1$ queue. Here, the influence of variations in service time are eliminated, and by [14, Thm. 8.6.2], we know that the workload in the $M/D/1$ queue is smaller than in the $M/M/1$ queue. (This result can also easily be established using the ideas of Sect. 4.) All three systems, the fluid approximation, the $M/M/1$ queue, and the $M/D/1$ queue, can be found in Figure 1.

In Figure 2, standard homogeneous $M/M/1$ and $M/D/1$ queues and their fluid approximation are plotted for the arrival rate 0.5 and mean service time 0.8. The nontrivial factor here is the presence of a large initial workload. We see, as in Figure 1, that the fluid model captures well the behavior of the queueing systems, as long as the stationary situation has not yet been reached.

Finally, in Figure 3, we consider a situation with two queues in tandem. It is a homogeneous system, where we assume a high initial load at the first queue and a service rate at the second queue equal to 0.6, lower than the service rate at the first

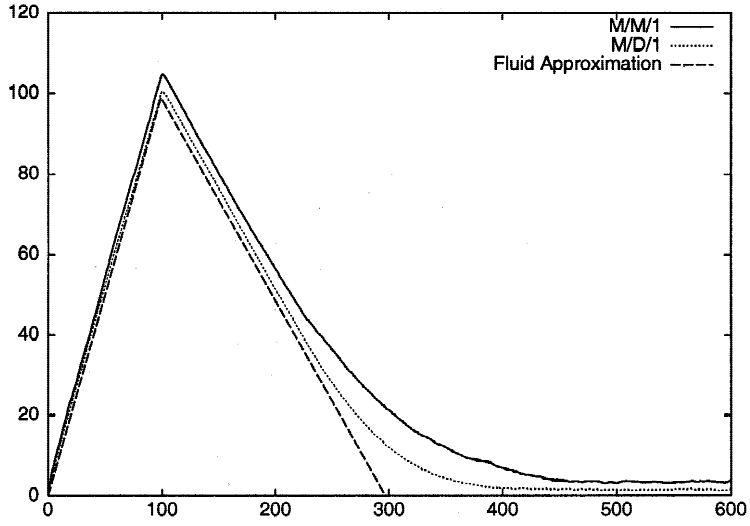

FIGURE 1. A single queue with a varying arrival rate.

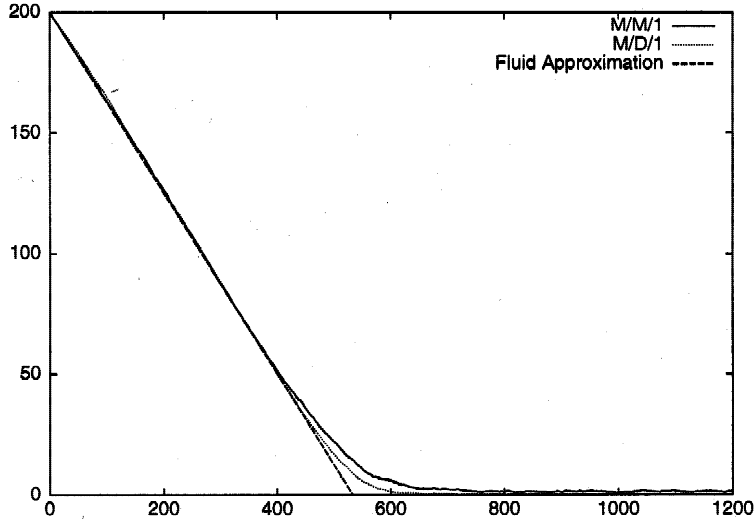

FIGURE 2. A single queue with a high initial workload.

queue (0.8) (the inverse would give fluid limit 0 for the second queue all the time). In Figure 3, the workloads at queues 1 and 2 are plotted, both for the fluid model and the exponential model, as well as the sum of the two queues for each model. For reasons of clarity, we did not add results for deterministic service times to the figure.

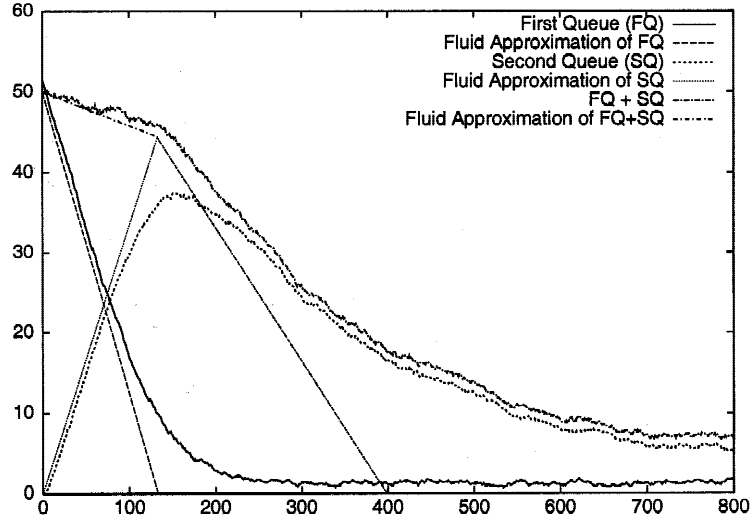

FIGURE 3. Tandem system with an initial workload and queue 1 faster than queue 2.

Of course, queue 1 has, on average, a higher workload than the corresponding fluid queue. It is interesting to note that this does not hold for queue 2, as we can see from Figure 3. This was to be expected, as $f(x) = x_2$ is not downstream decreasing. The function $f(x) = x_1 + x_2$ is downstream decreasing and directionally convex, and, indeed, we observe in Figure 3 that the total workload in the fluid system is majorized by the average total workload in the queueing system.

References

1. Altman, E. & Koole, G.M. (1998). On submodular value functions and complex dynamic programming. *Stochastic Models* 14: 1051–1072.
2. Asmussen, S. & Koole, G.M. (1993). Marked point processes as limits of Markovian arrival streams. *Journal of Applied Probability* 30: 365–372.
3. Chang, C.S., Chao, X., Pinedo, M., & Weber, R.R. (1992). On the optimality of LEPT and $c\mu$-rules for machines in parallel. *Journal of Applied Probability* 29: 667–681.
4. Chen, H. & Mandelbaum, A. (1991). Discrete flow networks: Bottleneck analysis and fluid approximations. *Mathematics of Operations Research* 16: 408–446.
5. Glasserman, P. & Yao, D.D. (1994). Monotone optimal control of permutable GSMPs. *Mathematics of Operations Research* 19: 449–476.
6. Kleinrock, L. (1975). *Queueing systems.* Vol. II: *Computer applications.* New York: Wiley.
7. Koole, G.M. (1995). *Stochastic scheduling and dynamic programming.* Amsterdam: CWI.
8. Koole, G.M. (1998). Structural results for the control of queueing systems using event-based dynamic programming. *Queueing Systems* 30: 323–339.
9. Koole, G.M. (1999). Convexity in tandem queues. Technical Report WS-516, Vrije Universiteit Amsterdam. Electronically available at www.cs.vu.nl/~koole/papers/WS516.ps.
10. Koole, G.M. & Liu, Z. (1998). Stochastic bounds for queueing systems with multiple on-off sources. *Probability in the Engineering and Informational Sciences* 12: 25–48.
11. Lin, W. & Kumar, P.R. (1984). Optimal control of a queueing system with two heterogeneous servers. *IEEE Transactions on Automatic Control*: AC-29: 696–703.
12. Meester, L.E. & Shanthikumar, J.G. (1993). Regularity of stochastic processes. *Probability in the Engineering and Informational Sciences* 7: 343–360.
13. Newell, G.F. (1971). *Applications of queueing theory.* London: Chapman & Hall.
14. Ross, S.M. (1983). *Stochastic processes.* New York: Wiley.
15. Schassberger, R. (1973). *Warteschlangen.* Berlin: Springer-Verlag.
16. Weber, R.R. & Stidham, S., Jr. (1987). Optimal control of service rates in networks of queues. *Advances in Applied Probability* 19: 202–218.