Available online at www.sciencedirect.com

SCIENCE DIRECT®

Operations Research Letters 32 (2004) 273–282

Operations Research Letters

www.elsevier.com/locate/dsw

Approximations for Markovian multi-class queues with preemptive priorities

Matthieu van der Heijden[a,*], Aart van Harten[a], Andrei Sleptchenko[b]

[a]Faculty of Business, Public Administration and Technology, Department of Operational Methods for Production and Logistics, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands

[b]EURANDOM, Eindhoven University of Technology, Netherlands

Received 1 September 2003; accepted 8 September 2003

Abstract

We discuss the approximation of performance measures in multi-class $M/M/k$ queues with preemptive priorities for large problem instances (many classes and servers) using class aggregation and server reduction. We compared our approximations to exact and simulation results and found that our approach yields small-to-moderate approximation errors.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Multi-channel queues; Preemptive priorities; Approximations

1. Introduction

Multi-class priority queues arise in various applications, for example in manufacturing systems [2]. We encountered this queueing model during our research on inventory control of repairable spare parts [10]. Then, repair shops can be modelled as multi-server queues, where each item type corresponds to a customer class with its own arrival and service process. To reduce the inventory of expensive spare parts, it might be worthwhile to reduce the repair throughput times of these items by giving them high priority. To be able to examine this, we need multi-class, multi-server priority queues to model repair shops. As key performance indicators, we need the first two moments of the number of items in the system per class.

Multi-class, multi-server queues with first-come-first-serve discipline have been studied before in [5,11,12]. Quite some attention has been given in the literature to single-class, multi-server queues with preemptive priorities, see e.g. [4,7,8]. As far as we know, [9] is the only paper dealing with the combination of preemptive priorities and multiple customer subclasses that are assigned either high or low priority. They provide an exact analysis and solution procedure for the steady-state probabilities. Therefore, they are able to derive a wide range of performance measures. A drawback is the computational effort required, so that large systems with many servers and many subclasses cannot be evaluated numerically. We need a less demanding procedure to deal with practical problems having many classes and servers.

* Corresponding author. Tel.: +31-53-489-2852; fax: +31-53-489-2159.
E-mail address: m.c.vanderheijden@sms.utwente.nl (M. van der Heijden).

0167-6377/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2003.09.001

To this end, we focus on faster approximation procedures for the multi-class, $M/M/k$ priority queue, based on exact results from [9]. First, we show in Section 2 that we can use class aggregation to speed up the computation of performance measures in the multi-class $M/M/k$ queue with FCFS discipline. We can use these approximations for the high-priority classes in the multi-class $M/M/k$ priority queue, because we focus on preemptive priorities, so that the performance of high-priority items does not depend on the presence of low-priority items. Next, we focus on approximations for the low-priority classes based on the analysis of a multi-class $M/M/1$ priority queue and using some correction factors (Section 3). We tested all our approximations in numerical experiments that we include in Sections 2 and 3. We found indications that our approximations provide reasonable to good results and that they can be used to analyse large models with many classes and servers. For example, we computed the performance characteristics of a 11 server, 23 class model within 90 s on a Pentium-II 700 MHz PC.

2. Multi-class *M/M/K* queue

2.1. Run times for exact analysis

For the high-priority classes, we can use the algorithm as given in [5] for the multi-class $M/M/k$ queue with FCFS discipline. They present a method to calculate the steady-state probabilities exactly. To judge the usefulness for large systems, we evaluated the run times for various combinations of number of servers and number of classes (the run times are fairly independent of other system parameters). In Table 1, we show the run times in seconds on a Pentium-II 700 MHz PC (all run times in the remainder of this paper refer to the same computer). We see that the algorithm can handle quite some combinations of the number of servers and number of classes within a reasonable amount of time. A wide range of cases can be analysed if either the number of servers or the number of classes is small (1, 2 or 3). The run time explodes however if both parameters become large. This is caused by the fact that eigenvalues and eigenvectors of a matrix of dimension $d(N,k)^{*}d(N,k)$ have to be computed, where $d(N,k)=2\binom{N+k-1}{k}$ with N the number of classes. Hence, we need approximations for large N and k.

2.2. Approximations

A common way to approximate k-server queues is by considering a single-server queue where the server works k times as fast and using some correction factor for scaling from k servers to a single server, see e.g. [3,14]. An alternative approach is to reduce the number of *classes* rather than the number of servers. That is, we derive the performance characteristics for class i by aggregating all other items into a single class. To obtain the performance characteristics for all classes $i=1,\dots,N$, we repeat this procedure N times for each item class. Then we have to analyse N two-class models rather than one N-class model. We see from Table 1 that this approach strongly reduces run times if N is large. As some preliminary experiments indicated

Table 1
Run times to solve the multiclass $M/M/k$ queue exactly (s)

Classes	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=15$	$k=20$	$k=25$	$k=50$
$N=2$	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	0.8
$N=3$	<0.1	<0.1	<0.1	<0.1	<0.1	0.1	0.3	0.5	0.9	11	91	573	
$N=4$	<0.1	<0.1	0.1	0.5	2	7	20	65	207				
$N=5$	<0.1	0.1	0.9	7	56	368							
$N=6$	<0.1	0.5	7	140	1452								
$N=7$	<0.1	2	58	1470									
$N=8$	0.1	7	390										
$N=9$	0.3	21											
$N=10$	0.6	78											

that server reduction may not yield as accurate results as class reduction, we decided to focus on the latter approach.

This still leaves us with the issue how to model an aggregate class. A simple approach is to assume that the aggregate class has an exponential service time distribution with a mean equal to the weighted mean service times of all items in the aggregate class. Then the aggregate item A has the following arrival rate λ_A and mean service time $E[S_A]$:

$$\lambda_A = \sum_{j \neq i} \lambda_j \quad \text{and} \quad E[S_A] = \frac{\sum_{j \neq i} \lambda_j E[S_j]}{\sum_{j \neq i} \lambda_j},$$

where λ_j and $E[S_j]$ denote the arrival rate and the mean service time of class j customers, respectively.

However, it is clear that the service time of the aggregate item is generally not exponentially distributed. If the service rates of all items being aggregated are very different, the aggregate service time distribution is not even close to an exponential one. Therefore, we will also use a hyperexponential distribution for the service time of the aggregate item, defined as

$$f_A(t) = p v_1 e^{-v_1 t} + (1-p) v_2 e^{-v_2 t}. \qquad (1)$$

The resulting two-class queueing system is equivalent to a three-class $M/M/k$ queue (see e.g. [11]) with respective arrival rates and mean service times

$$\lambda_i \quad \text{and} \quad E[S_i],$$

$$\lambda_{A1} = p \sum_{j \neq i} \lambda_j \quad \text{and} \quad E[S_{A1}] = 1/v_1,$$

$$\lambda_{A2} = (1-p) \sum_{j \neq i} \lambda_j \quad \text{and} \quad E[S_{A2}] = 1/v_2.$$

We choose the parameters p, v_1 and v_2 of the hyperexponential distribution (1) such that the first three moments coincide with the first three moments of the aggregate service time distribution. That is, we choose (cf. [1])

$$v_1 = \frac{1}{2}\left(a_1 + \sqrt{a_1^2 - 4a_2}\right),$$

$$v_2 = \frac{1}{2}\left(a_1 - \sqrt{a_1^2 - 4a_2}\right),$$

$$p = 1 - \frac{v_2(E[S_H]v_1 - 1)}{v_1 - v_2}, \qquad (2)$$

where

$$a_2 = \frac{6E[S_H] - (3E[S_H^2]/E[S_H])}{(6\{E[S_H^2]\}^2/4E[S_H]) - E[S_H^3]} \quad \text{and}$$

$$a_1 = \frac{1}{E[S_H]} + \frac{a_2 E[S_H^2]}{2E[S_H]}.$$

For exponentially distributed class service times, the nth moment of the aggregate service time equals

$$E[S_A^n] = \frac{n! \sum_{j \neq i} \lambda_j (E[S_j])^n}{\sum_{j \neq i} \lambda_j}.$$

It is known that the following two conditions are necessary to find valid parameters as specified by (2) cf. [13]:

1. the coefficient of variation of the aggregate service time c_A (the ratio of the standard deviation and the mean) should be at least 1; it can easily be shown that this condition always holds for a weighted average of exponentially distributed random variables as in this application;
2. the third moment of the aggregate service time should satisfy $E[S_A^3] \geqslant \frac{3}{2}(1 + c_A^2)^2 \{E[S_A]\}^3$; it is not guaranteed that this lower bound is always satisfied; if it is not, we choose the parameters of the hyperexponential distribution such, that the third moment equals the lower bound, thereby approximating the aggregate service process as close as possible.

2.3. Numerical results

In a numerical experiment, we examined the quality of two approximations for the multi-class $M/M/k$ queue by comparison to exact results using the method as given in [5]. We computed the performance characteristics of a specific class i by aggregating all other items in a single class having either an exponential distribution (M-approximation) or a hyperexponential distribution (H_2-approximation). So, to analyse an N-class $M/M/k$ queue, we computed the performance of either N two-class $M/M/k$ queues or N three-class $M/M/k$ queues. We expected that the second approach would be more accurate, but also more time consuming. Therefore, we considered both the relative approximation error and the run times.

In all our experiments, we considered a six-class $M/M/k$ queue. As normalisation, we chose for the total arrival rate $\sum_i \lambda_i = 1$. We varied the number of servers ($k = 3$ or 6) and the utilisation ($\rho = 0.60$ or 0.90). Further, we chose three scenarios for the differences between classes:

1. all classes have the same arrival rate and the service rates $\mu_i = 1/E[S_i]$ decrease geometrically such, that the ratio of the highest to the lowest service rate equals 10;
2. both the arrival rates and the service rates decrease geometrically such, that the ratio of the highest to the lowest rate equals 10;
3. the arrival rates decrease and the service rates increase such, that the ratio of the highest to the lowest rate equals 10.

So we considered quite extreme cases, including a few where one item requires 100 times as much server capacity as another one. We chose to include such extreme cases, because we expected that the quality of our approximations might deteriorate then.

As performance measures, we focused on the mean and coefficient of variation of the number of items in the system per class (denoted by N_i), because that is what we need for our application (to model repair shops in spare part inventory models). Besides, we chose the mean queue length per class $E[Q_i]$ as a generally relevant performance characteristic. Note that the difference between $E[N_i]$ and $E[Q_i]$ is exactly the mean number of class i items in service. In Table 2, we show the average and maximum relative approximation error (%) over all classes for each case.

We see that the M-approximation error is very high, whereas the H_2-approximation yields much better results. A probable cause is that the aggregate service time distribution is not even close to exponential in the experiment, so that the inclusion of the third and second moment has a strong added value. For the run times, we found that the exact solution takes about 0.5 s if $k = 3$ and about 24 min if $k = 6$. The approximations were all much faster, namely 0.02 and 0.37 s worst case for the M-approximation and the H_2-approximation, respectively. Based on these results, we recommend the H_2-approximation.

Remarks. (1) A limitation is that we are not able to calculate *correlations* between the number of items in queue or in the system per class. For pairwise correlations, we can resolve this by isolating classes i and j while aggregating all other items in a single class. Then we can apply the method by Harten and Sleptchenko [5] to find the correlations between the

Table 2
Average and maximum relative approximation errors (%) for multi-class $M/M/k$ queues

Perf. measure	Error	Approx.	Equal arrival rates				Decreasing arrival rates							
			Service rates decrease				Service rates decrease				Service rates increase			
			$\rho = 0.60$		$\rho = 0.90$		$\rho = 0.60$		$\rho = 0.90$		$\rho = 0.60$		$\rho = 0.90$	
			$k = 3$	$k = 6$	$k = 3$	$k = 6$	$k = 3$	$k = 6$	$k = 3$	$k = 6$	$k = 3$	$k = 6$	$k = 3$	$k = 6$
$E[Q_i]$	Av	M	25.4	23.0	27.6	27.1	31.1	27.0	34.8	34.0	13.4	12.2	14.6	14.4
		H_2	0.3	0.5	0.1	0.1	0.6	1.1	0.1	0.2	0.1	0.2	0.0	0.0
	Max	M	31.4	28.3	34.4	33.8	38.0	32.7	43.0	41.2	17.2	15.6	18.8	18.5
		H_2	0.4	0.8	0.1	0.1	0.8	1.5	0.2	0.3	0.1	0.3	0.0	0.0
$E[N_i]$	Av	M	10.1	3.7	23.1	19.3	9.2	2.8	26.9	21.1	6.2	2.5	12.7	10.9
		H_2	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.0	0.0	0.0
	Max	M	15.6	6.9	28.3	24.1	15.0	4.8	36.0	30.5	10.5	5.2	15.8	14.7
		H_2	0.2	0.2	0.1	0.1	0.3	0.2	0.1	0.2	0.1	0.1	0.0	0.0
$c[N_i]$	Av	M	1.0	0.3	1.3	2.9	1.9	1.2	2.6	5.4	1.9	0.8	2.3	2.4
		H_2	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.0
	Max	M	2.4	0.4	2.5	6.0	5.6	4.1	4.3	9.2	4.7	2.1	5.4	4.6
		H_2	0.1	0.1	0.1	0.1	0.3	0.0	0.1	0.2	0.0	0.0	0.0	0.0

classes i and j using a four-class approximation (in case of the H_2-approach). We need $\frac{1}{2}N(N-1)$ runs of a four-class $M/M/k$ queue to find all pairwise correlations. Although four-class models consume more time, we can easily analyse models up to $k=10$ servers (see Table 1).

(2) We can approximate multi-class $M/H_n/k$ queues analogously, i.e. by class aggregation. Again, we need to analyse four-class $M/M/k$ queues, which is manageable up to $k=10$ servers. For more servers, we should move to the M-approximation. Note that we cannot approximate other phase-type service distributions, such as Erlang distributions, because a sequence of exponential phases does not fit in the multi-class $M/M/k$ model.

3. Multi-class *M/M/k* queue with preemptive priorities

3.1. Run times for exact analysis

For multi-class queues with two priority groups and preemption, we proceed from the exact algorithm by Sleptchenko et al. [9]. First, we examine the run time requirements, see Table 3. We see that the exact algorithm is only useful for small problems and for cases with either $k \leqslant 2$ or only one high-priority ($n_\mathrm{h}=1$) and one low-priority class ($n_\mathrm{l}=1$), otherwise the computation time explodes. In an exact approach, the computation now requires ± 20 matrix inversions of dimension

$$\sum_{i=0}^{k}\left[\sum_{j=i}^{k}\binom{n_\mathrm{h}+j-1}{j}\binom{n_\mathrm{l}+k-j-1}{k-j}\right]\times\binom{n_\mathrm{l}+i-1}{i}$$

using LR-decomposition, see [9]. Because of these run time limitations we can build our approximations only upon the special cases, $k \leqslant 2$ or $(n_\mathrm{h}, n_\mathrm{l})=(1,1)$.

3.2. Approximations

For priority queues with preemption, we have to focus on low-priority items only, because we can use the results from the previous section for high-priority items. Although class aggregation appeared to be useful for the multi-class $M/M/k$ queue with FCFS, it does not make sense to use a similar approach for priority queues because of still excessive run times. To approximate the performance of low-priority item i, we have to aggregate all high-priority items into a single class and all remaining low-priority items in another single class. That is, we get a model with two high-priority items and three low-priority items for the H_2-approximation. It is clear from Table 3 that run times then already become unmanageable for $k>3$. The M-approximation yields a model with one high-priority item and two low-priority items, which is only useful for a moderate number of servers, say $k \leqslant 6$ (see Table 3). For more general cases, we need an alternative approach.

Therefore, we move to server reduction as proposed by Buzen and Bondi [3]. Let us introduce the following notation:

$EQ_i(PR, n_\mathrm{h}, n_\mathrm{l}, \bar{\mu}, k)$

= mean number of class i items in the queue for a multi-class priority queue with k servers, n_h high-priority classes, n_l low-priority classes and service rate vector $\bar{\mu}$;

Table 3
Run times to solve the multiclass $M/M/k$ queue with preemptive priorities exactly (s)

Classes (high, low priority)	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=12$	$k=15$	$k=20$
$(n_\mathrm{h}, n_\mathrm{l})=(1,1)$	<0.1	<0.1	0.1	0.2	0.6	1.5	3	7	11	20	56	187	1067
$(n_\mathrm{h}, n_\mathrm{l})=(1,2)$	<0.1	0.3	3	26	153	765							
$(n_\mathrm{h}, n_\mathrm{l})=(1,3)$	<0.1	1.5	39	769									
$(n_\mathrm{h}, n_\mathrm{l})=(2,2)$	0.1	3	65	1421									
$(n_\mathrm{h}, n_\mathrm{l})=(2,3)$	0.1	15	995										

$$EQ_i(FCFS, n_{\mathrm{h}}, n_{\mathrm{l}}, \bar{\mu}, k)$$
$$= \text{mean number of class } i \text{ items in the queue for a multi-class FCFS queue with } k \text{ servers, } n_{\mathrm{h}} \text{ high-priority classes, } n_{\mathrm{l}} \text{ low-priority classes and service rate vector } \bar{\mu}.$$

A similar notation applies for other performance characteristics like the mean number of type i low-priority items in the system N_i, etc. For the mean waiting time $E[W]$ in a single-class $M/M/k$ preemptive priority queue, [3] propose to replace k servers by a single server that works k times as fast and to use a correction factor, being the ratio of the waiting times when the same trick would be applied to the non-priority multi-server queue

$$EW(PR,1,1,\bar{\mu},k) \approx EW(PR,1,1,k\bar{\mu},1)^* \frac{EW(FCFS,1,1,\bar{\mu},k)}{EW(FCFS,1,1,k\bar{\mu},1)}.$$

Extending this approach to the multi-class priority system and applying it to other performance measures, we get the approximation

$$EQ(PR,n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k) \approx EQ(PR,n_{\mathrm{h}},n_{\mathrm{l}},k\bar{\mu},1)^* C_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k),$$

where $C_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k)$ denotes the correction factor for the mean number of class i items in the queue if we approximate a preemptive priority queue with n_{h} high-priority classes, n_{l} low-priority classes and k servers by an equivalent single server queue having a server that works k times as fast. We consider three options for this correction factor (A, B and C). First, a straightforward extension of the correction factor from [3] yields

$$C^A_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k) = \frac{EQ_i(FCFS,n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k)}{EQ_i(FCFS,n_{\mathrm{h}},n_{\mathrm{l}},k\bar{\mu},1)}.$$

To calculate the mean waiting time in a multi-class $M/M/k$ queue with k servers (numerator of the correction factor), we use the H_2-approximation from the previous section to avoid excessive run times.

Correction factor A does not take into account the fact that the server capacity that is available for low-priority classes may fluctuate strongly in time. Therefore, we consider another correction factor that can be evaluated quickly and that is based on priority queues. We aggregate all high-priority items into a single class with an exponentially distributed service time and we do the same for all low-priority items. We denote the resulting two-dimensional aggregate service rate vector by $\tilde{\mu}$. For this two-class model, we calculate the correction factor for scaling with the number of servers k as follows:

$$C^B_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k) = \frac{EQ_i(Pr,1,1,\tilde{\mu},k)}{EQ_i(Pr,1,1,k\tilde{\mu},1)}.$$

Because we aggregate all low-priority items, correction factor B is equal for all items. Because the low-priority classes may have completely different characteristics, we possibly need different correction factors. Therefore, we combine the advantage of correction factor A (distinguish between classes) and correction factor B (take into account impact of high-priority items) in correction factor C

$$C^C_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k) = C^B_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k)^* \frac{C^A_{Q,i}(n_{\mathrm{h}},n_{\mathrm{l}},\bar{\mu},k)}{C^A_{Q,i}(1,1,\tilde{\mu},k)}. \quad (3)$$

By the ratio of the correction factor for multiple high- and low-priority classes and the correction factor for an aggregate high- and an aggregate low-priority item, we adjust for class differences.

As an approximation for the multi-class $M/M/k$ queue with preemptive priorities, we propose to use a combination of class reduction and server reduction as described above using one of the correction factors A, B or C. We apply these correction factors to

- $E[Q_i]$, the mean number of class i items per class in the queue excluding the postponed items (i.e. the low-priority items that are preempted by high-priority items);
- $E[P_i]$, the mean number of postponed class i items;
- $Var[N_i]$, the variance of the number of items in the system.

We do *not* apply the correction factor directly to the mean number of class i items in the system, $E[N_i]$. Some preliminary experiments showed us that it is better to approximate the three components of $E[N_i]$ separately, namely the mean number of items in the queue $E[Q_i]$, being postponed $E[P_i]$ and in service

$E[R_i]$ ($=\lambda_i E[S_i]$). This approach yields a better approximation, because the correction factors of $E[Q_i]$ and $E[P_i]$ appear to be completely different. In principle, we can make a similar factorisation of $Var[N_i]$ in the variances of Q_i, P_i and R_i and their mutual correlations, yielding totally six components and thus six correction factors. Because of simplicity, we decided to apply the correction factor directly to $Var[N_i]$.

3.3. Numerical results

First, we compared our approximations to exact results using the method from [9]. We have to restrict ourselves to relatively small cases, because the exact method is too time consuming otherwise (see Table 3). Therefore, we considered one model variant with two high-priority classes, three low-priority classes and $k=3$ servers, and another model variant with one high-priority class, three low-priority classes and $k=4$ servers. For each variant, we varied the total utilisation ($\rho=0.7$ or 0.9), the utilisation for high-priority items only ($\rho_H=0.3$ or 0.6), the ratio of the overall service rates of high- and low-priority items ($\gamma=\mu_L/\mu_H=0.5$ or 2) and three scenarios for the differences between classes (equal for high- and low-priority class):

1. All classes have the same arrival rate and the service rates decrease geometrically such, that the ratio of the highest to the lowest service rate equals 5.
2. Both the arrival rates and the service rates decrease geometrically such, that the ratio of the highest to the lowest rate equals 5.
3. The arrival rates decrease and the service rates increase such, that the ratio of the highest to the lowest rate equals 5.

As normalisation, we set the total arrival rate equal to 1. In total, we considered $2^4 \times 3 = 48$ cases. For each case, we calculated the mean and maximum relative approximation error over all low-priority classes for$E[Q_i]$, $E[N_i]$ and $c[N_i]$ using the correction factors A, B and C. The key results are summarised in Table 4. The run times for the exact calculations varied between 14 and 46 min/case, whereas the approximations always took less than 1 s/case for each correction factor.

We see that correction factor C yields the best results. Apparently, it is worthwhile to take into account both the variation in available server capacity for low-priority items and the differences between low-priority classes. We also note that the approximation error is smallest for the mean number of items in the system, probably because we know one component (the mean number of items in service) exactly. The approximation errors are considerably larger for the coefficient of variation of the number of items in the system, although they are still within reasonable bounds. To find the causes of the errors, we analysed the errors on a factor-by-factor basis, see Table 5.

Table 4
Average and maximum relative errors (%) using the three correction factors

Correction factor	$E[Q_i]$		$E[N_i]$		$c[N_i]$	
	Av.	Max.	Av.	Max.	Av.	Max.
A	4.8	19.1	12.6	43.7	6.0	15.3
B	2.5	13.9	1.1	5.4	4.2	20.9
C	1.1	6.7	0.6	2.8	1.9	11.4

We found that the approximation error especially increases if the total utilisation decreases and the utilisation of high-priority items increases, which is similar to the findings by Buzen and Bondi [3] and Whitt [14]. Also, there is an indication that the error increases with the number of servers (compare the model variants 1 and 2).

To validate our finding that our approximations using correction factor C as in (3) yield reasonable results, we used discrete event simulation. We constructed a model in the object oriented simulation tool eM-Plant. Unfortunately, we could run a limited number of cases only, because our simulation model required much computation time until convergence, i.e. until the 95% confidence interval for the mean variance of the number of low-priority items in the system was smaller than 2% (between several hours and several days per case; in a few cases, we cut of the simulation because of excessive run time).

We selected a case from the literature to test our approximations, namely the repair shop model by Hausman and Scudder [6]. They consider engines at a commercial airline that consist of 23 components (item classes) that are repaired by a 10-server repair shop, see Table 6 for the component data. We

Table 5
Influence of the experimental factors on the approximation error (%) for correction factor C

Experimental factor	$E[Q_i]$		$E[N_i]$		$c[N_i]$	
	Av.	Max.	Av.	Max.	Av.	Max.
Model variant 1 ($n_h = 2$, $n_l = 3$, $k = 3$)	0.7	3.3	0.5	2.8	1.6	7.6
Model variant 2 ($n_h = 1$, $n_l = 3$, $k = 4$)	1.6	6.7	0.7	2.8	2.2	11.4
$\rho = 0.7$	1.7	6.7	0.8	2.8	2.8	11.4
$\rho = 0.9$	0.5	1.8	0.4	1.7	1.1	6.6
$\rho_H = 0.3$	0.7	2.6	0.3	1.4	1.5	6.2
$\rho_H = 0.5$	1.6	6.7	0.8	2.8	2.3	11.4
$\gamma = 0.5$	1.4	6.7	0.6	2.8	1.4	7.0
$\gamma = 2$	0.9	3.3	0.6	2.8	2.4	11.4
Arrival and service rate scenario 1	1.1	5.2	0.5	2.8	1.8	8.4
Arrival and service rate scenario 2	1.8	6.7	0.9	2.8	2.4	11.4
Arrival and service rate scenario 3	0.5	2.6	0.4	2.0	1.5	7.0

Table 6
Data from [6] that we used as test case

Component i	Number	λ_i	$E[S_i]$
C44	1	0.055560	8.46
C51	2	0.035552	8.60
C41	3	0.011112	10.60
C13	4	0.024197	14.59
C11	5	0.024197	14.10
C31	6	0.012825	17.40
C42	7	0.044448	11.15
C54	8	0.008888	15.95
C36	9	0.028850	10.63
C33	10	0.019225	15.45
C37	11	0.006413	14.60
C34	12	0.028850	13.15
C35	13	0.025638	18.16
C52	14	0.017776	14.55
C32	15	0.003200	18.00
C53	16	0.026664	18.00
C21	17	0.013882	14.98
C55	18	0.022220	21.64
C22	19	0.024990	25.05
C14	20	0.018820	29.07
C23	21	0.002778	16.08
C43	22	0.027780	24.95
C12	23	0.016137	27.70

constructed 15 runs based on these case data, where we varied the distribution of items over the high- and low-priority groups ($n_h = 5$, 8, 11, 14 and 17, where the first n_h items from Table 5 had high priority) and the number of servers ($k = 9$, 10 and 11). As a consequence, the total utilisation varied between $\rho = 0.71$ and 0.87 and the utilisation for high-priority items varied between $\rho_H = 0.14$ and $\rho_H = 0.55$. We calculated $E[N_i]$, $c[N_i]$ and $E[Q_i]$ using our approximate method with correction factor C as in (3). The run times for the 23-class priority queues with 9–11 servers varied between 30 and 90 s. We show the approximation errors, this time defined as the relative difference between approximated and simulated values, in Table 7.

We found larger errors than for the smaller cases in Table 4, as we already expected. Still the errors were within reasonable bounds for most cases. Note that a few high relative errors for the mean number of items in the queue occurred for a specific item class. For example, we encounter a maximum error of 25% if $(k, n_h, n_l) = (11, 5, 18)$. However, the mean number of items in the queue is about 0.005 for this case, which is almost zero. Hence the maximum relative error is large, but not very important and it may also be due to simulation errors. Of course, it is impossible to draw general conclusions based on a few large cases only. Still, these results indicate that our approximations can be useful to analyse large multi-class preemptive priority queueing systems, i.e. with many classes and servers.

Table 7
Average and maximum relative errors (%) for the Hausman and Scudder case (1982)

k	n_h	n_l	$E[Q_i]$		$E[N_i]$		$c[N_i]$	
			Average (%)	Maximum (%)	Average (%)	Maximum (%)	Average (%)	Maximum (%)
11	5	18	4.2	25.0	1.0	4.7	1.9	4.1
	8	15	6.3	11.4	0.8	3.6	3.2	5.4
	11	12	4.9	8.9	0.6	1.2	2.2	5.9
	14	9	3.7	5.4	0.7	1.8	5.9	8.9
	17	6	4.2	7.5	0.6	1.4	6.5	10.0
10	5	18	3.0	6.8	0.9	3.2	0.8	1.7
	8	15	4.3	8.3	1.0	2.7	0.9	2.0
	11	12	0.9	3.0	0.8	3.2	1.3	3.6
	14	9	2.4	4.4	0.5	1.1	1.6	4.7
	17	6	3.1	3.1	0.6	1.3	1.9	6.5
9	5	18	0.4	1.0	0.3	0.6	2.5	3.7
	8	15	0.5	1.8	0.3	0.9	3.5	5.5
	11	12	2.6	4.2	1.1	2.2	4.7	7.4
	14	9	2.3	3.1	1.0	1.4	4.8	7.0
	17	6	2.3	3.2	0.8	1.3	4.3	6.1
Total			3.0	25.0	0.7	4.7	3.1	10.0

Remarks. 1. Our approach is also suitable to analyse models with more than two preemptive priority classes. When approximating the performance characteristics of all items in a certain priority class m, we ignore all lower-priority items and we aggregate all higher-priority items in a single priority class with hyperexponential service times as proposed in Section 2. Working from the highest to the lowest-priority class, we can find approximations for the performance characteristics of all items in all priority classes. Such an approach has been proposed many times before, see e.g. [8]. However, they proposed to aggregate all items to a single item with exponential service time distribution. As we showed in Section 2, it is better to include higher moments of the service time of the aggregate item.

2. Extension of our approach to hyperexponential service time distributions and/or correlations between item classes is not as straightforward as for the FCFS multi-class queue because of run time limitations. Further research is required in this respect.

References

[1] T. Altiok, On the phase type approximation of general distributions, IIE Trans. 17 (2) (1985) 110–116.
[2] J.A. Buzacott, J.G. Shanthikumar, Stochastic Models of Manufacturing Systems, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[3] J.P. Buzen, A.B. Bondi, The response times of priority classes under preemptive resume in M/M/m queues, Oper. Res. 31 (3) (1983) 456–465.
[4] H.R. Gail, S.L. Hantler, B.A. Taylor, On preemptive Markovian queue with multiple servers and two priority classes, Math. Oper. Res. 17 (2) (1992) 365–391.
[5] A. van Harten, A. Sleptchenko, On multi-class, multi-server queuing, Queueing Systems 43 (4) (2003) 307–328.
[6] W. Hausman, G. Scudder, Priority scheduling rules for repairable inventory systems, Manage. Sci. 28 (1982) 1215–1232.
[7] E.P.C, Kao, S.D. Wilson, Analysis of nonpreemptive priority queues with multiple servers and two priority classes, European J. Oper. Res. 118 (1999) 181–193.
[8] I. Mitrani, P.J.B. King, Multiprocessor systems with preemptive priorities, Performance Evaluation 1 (1981) 118–125.
[9] A. Sleptchenko, A. van Harten, M.C. van der Heijden, Analyzing multi-class, multi-server queueing systems with preemptive priorities, Working Paper, Faculty of Technology and Management, University of Twente, Enschede, the Netherlands, 2002, submitted for publication.

[10] A. Sleptchenko, M.C. van der Heijden, A. van Harten, Using repair priorities to reduce stock investment in spare part networks, Working paper, Faculty of Technology and Management, University of Twente, Enschede, the Netherlands, 2002, submitted for publication.

[11] J.H.A. de Smit, The queue *GI*/*M*/*s* with customers of different types or the queue $GI/H_m/s$, Adv. Appl. Probab. 15 (1983) 392–419.

[12] J.H.A. de Smit, A numerical solution for the multi-server queue with hyper-exponential service times, Oper. Res. Lett. 2 (5) (1983) 392–419.

[13] W. Whitt, Approximating a point process by a renewal process I: two basic methods, Oper. Res. 30 (1982) 125–147.

[14] W. Whitt, Approximations for the *GI*/*G*/*c* queue, Production Oper. Manage. 2 (1993) 144–161.