Annals of Operations Research 93 (2000) 15–40

AMVA-based solution procedures for open queueing networks with population constraints

Ronald Buitenhek [a,*], Geert-Jan van Houtum [b] and Henk Zijm [b]

[a] *Faculty of Mechanical Engineering, University of Twente, Enschede, The Netherlands*
[b] *Faculty of Technology Management, Eindhoven University of Technology, Eindhoven, The Netherlands*

We propose a new method for the performance evaluation of Open Queueing Networks with a Population Constraint (represented by a set of tokens). The method is based on the application of Approximate Mean Value Analysis (AMVA) algorithms. We present procedures for single class networks and for multiple class networks, subject to either a common constraint (shared tokens) or to class-based constraints (dedicated tokens). In fact, the new method is a unified framework into which all procedures for the different types of networks fit. We show how the new method relates to well-known methods and present some numerical results to indicate its accuracy.

1. Introduction

We consider the performance evaluation of *Open Queueing Networks with a Population Constraint* (OQNs-PC). In a single job class OQN-PC the total number of jobs in service or waiting for service at the set of service stations is never larger than the maximum number N. The set of service stations constitute the *inner system.* Jobs that arrive when the inner system is full are forced to wait outside in an *external queue*. Once a job is inside the inner system, it follows its Markovian routing until it has completed its final operation. Upon the departure of a job, the job in the first position of the external queue, if any, is allowed to enter the inner system. Due to the presence of the external queue, the OQN-PC fundamentally differs from a loss system, in which arriving jobs that find the network full are lost.

The population constraint of the inner system can be represented by a set of *tokens*. To be allowed in the inner system a job must first get a token. The job keeps the token during its stay in the inner system and releases it upon departure.

An obvious generalization of the single class OQN-PC is the multiclass OQN-PC. For the multiclass OQNs-PC, we distinguish three cases. Either each job class has a *dedicated population constraint* represented by a set of class-specific tokens, or the classes have a *common population constraint* represented by a set of tokens that

* Corresponding author. Current address: KPN Research, P.O. Box 421, 2260 AK Leidschendam, The Netherlands. E-mail: r.buitenhek@research.kpn.com.

are shared by the classes, or a *combination* occurs in which population constraints are dedicated to disjoint subsets of classes.

The OQN-PC can be used for the performance evaluation of manufacturing systems. Consider, for example, a Flexible Manufacturing System (FMS) in which jobs need to be clamped on a pallet while they are processed at the work stations. The work stations are then modeled by the service stations and the pallets are modeled by the tokens. FMSs are often modeled by Closed Queueing Networks (CQNs). A CQN model assumes that all pallets are always occupied and that whenever a part is unloaded a new part can always be loaded. Thus, using CQN models one may estimate the maximum number of parts that can be produced per time unit by the FMS (the maximum throughput). The OQN-PC model allows one to estimate the mean manufacturing lead time of a part, including the time it waits for a pallet, and to estimate the mean amount of work in process, including the parts waiting for a pallet. If the tokens are just a representation of the population constraint, the OQN-PC can be used to model manufacturing systems in which workload control is applied. This workload control aims at keeping the amount of work in process in the manufacturing system below a certain level. For pull systems, this control is referred to as the CONWIP control; see Spearman et al. [18].

The behavior of the *single class* OQN-PC with *exponential* servers and a Poisson arrival process is described by a Markov process with the states $(n_E, \vec{n}) = (n_E, n_1, \ldots, n_M)$, where n_E denotes the number of jobs in the external queue, M denotes the number of service stations and n_m the number of jobs at station m, $m = 1, \ldots, M$. Furthermore, $n_E = 0$ if $\sum_{j=1}^{M} n_j < N$. It can be easily seen that this Markov process is a Quasi Birth and Death (QBD) process: there can only be transitions from a state $(n_E, n_1, \ldots, n_M)$ to states with the number in the external queue equal to $n_E - 1$, n_E or $n_E + 1$. Unfortunately, the stationary distribution of the Markov process does not possess a product-form solution. However, the Markov process can be solved by numerical methods. One may use the matrix-geometric approach (see [14]) or aggregation–disaggregation techniques (see [17,19]), if the number of work stations M and the number of tokens N are not too large. For larger values of M and N, approximate solution procedures have been developed to obtain the desired performance measures.

A well-known approximate solution procedure for the OQN-PC with general servers is the *aggregation method* as already proposed by Avi-Itzhak and Heyman [1] in 1972. They considered an inner system with a special structure modeling a computer system. Buzacott and Shanthikumar [7] allowed a more general inner system and suggested to use this method for the performance evaluation of FMSs. This method basically considers the aggregated Markov process with states $n = n_E + n_I$, where $n_I = n_1 + \cdots + n_M \leqslant N$. The aggregated process is a birth and death process. Its upcrossing rates are all equal to the external arrival rate and for each state $n \geqslant 1$, the downcrossing rate is estimated by the throughput of the CQN with population $n_I = \min(n, N)$ that is constructed from the OQN-PC by properly redirecting the external departures. If there are non-exponential servers, approximate Mean Value

Analysis algorithms can be used to estimate the required throughput. Generalizations of the aggregation method to networks with multiple job classes and shared tokens are described in [6]. The direct generalization for the case with dedicated tokens yields a multidimensional random walk that is difficult to solve. Brandwajn [3], Lazowska and Zahorjan [12], and Thomasian and Bay [20] propose generalizations that involve extra approximations to keep the computational effort tractable.

Another approximation method for the single class OQN-PC with general servers is the *method of Dallery* [11], which has been proposed in 1990. This method is based on the idea that the OQN-PC is equivalent to a CQN if the roles of the jobs and the tokens are interchanged. The equivalent CQN has a population equal to the number of tokens and consists of the same stations as the OQN-PC plus an extra station modeling the external queue. The extra station is called a synchronization station, because there the jobs and the tokens synchronize. This CQN is then analyzed by an extension of the method of Marie (see [13]) as follows. The synchronization station and all non-exponential server stations are, in an iterative procedure, replaced by load-dependent exponential servers. The resulting product-form CQN is solved to obtain the internal queue lengths. The number of jobs in the external queue is approximated from a birth and death process with states denoting the number of jobs minus the number of tokens present at the synchronization station. The upcrossing rates of the birth and death process are equal to the external arrival rate of the jobs and the downcrossing rates are equal to the state-dependent arrival rate of the tokens as obtained by Marie's method. The method of Dallery and the aggregation method yield identical results if the inner system of the OQN-PC consists of only exponential servers. In that case, the birth and death processes as obtained by the two methods are identical. A generalization for the multiclass OQN-PC with dedicated tokens is described in Perros et al. [15], albeit that they restrict themselves to networks with a BCMP-type inner system. Another way to generalize the method of Dallery for multiclass networks is by the method as described by Baynat and Dallery [2]. Baynat and Dallery propose an extension of the method of Marie for very general multiclass *closed* queueing networks. Their method not only allows general service times, but also some versions of state-dependent routing, fork and join mechanisms and so-called synchronizations. When applied to the OQN-PC, an extra approximation is applied compared to the method of Dallery, to deal with the presence of multiple job classes. This extra approximation concerns the decomposition of an R-class network into R single-class networks. The decomposition allows a quick evaluation of networks with many classes, but may be expected to be less accurate than the method as proposed by Perros et al.

In this paper, we propose a third method. This new method is a *unified framework* that can be applied to all types of OQNs-PC. In this method, we adopt the interchanging of the jobs and the tokens as done by Dallery, but we apply Approximate Mean Value Analysis (AMVA) algorithms to solve the resulting CQN. For each of the different types of OQNs-PC, the precise procedure that has to be followed is described below (see sections 2–4). The contributions of this paper are the following. First of all, the main contribution is that the new method is applicable for all types of OQNs-PC.

The second contribution is as follows. For single-class OQNs-PC, the new method appears to be equivalent to the aggregation method. However, since the new method is formulated in a different way, it provides a *new view* of the aggregation method, which clearly shows the relation between the aggregation method and Norton's theorem and which allows a better qualitative comparison between the aggregation method and the method of Dallery. Third, to our knowledge, our description of the method for the multiclass OQN-PC with general purpose pallets is the first one in the open literature. Fourth, for the multiclass OQN-PC with dedicated pallets, the new method allows the analysis of networks with non-exponential service stations, without introducing an extra approximation as in the method described by Baynat and Dallery.

The paper is organized as follows. In section 2, we present the single-class model and the new method for this model. Section 3 and section 4 deal with OQNs-PC with multiple job classes. In section 3, we generalize the new method to the case with shared tokens, while in section 4, further generalization to the case with dedicated tokens is presented. Finally, in section 5, we discuss further possible extensions.

2. The single-class open queueing network with a population constraint

2.1. The queueing network

Consider the OQN-PC with a single job class. A schematic representation of the single-class OQN-PC, with the token representation of the population constraint, is given in figure 1. The OQN-PC can be decomposed into two subsystems: the *external queue* and the *inner system*. The external queue regulates the capacity constraint of the network or availability of the tokens. It consists of a queue for the jobs and a queue for the tokens. The inner system consists of the M service stations, numbered from 1 to M. Each service station may consist of multiple parallel servers. Furthermore, one may distinguish two flows of entities: the flow of jobs and the flow of the N tokens. The jobs arrive at the network according to a Poisson process with rate λ. A job moves through the network according to a Markovian routing, characterized by the routing matrix P. At service station m a job receives service according to a general

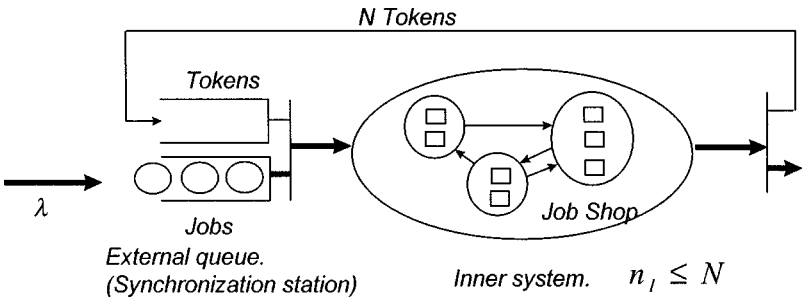

Figure 1. The open queueing network with a population constraint represented by a set of tokens.

distribution with known first two moments ES_m and ES_m^2. At each station, the jobs are served in a first-come first-served order.

The regulation of the population constraint is as follows. If upon arrival a job finds a token in the external queue, the job and the token synchronize and together enter the inner system. Otherwise, the job joins the external queue at the back. Upon an external departure of a job, the token that was synchronized with that job returns to the external queue. The first in line job in the external queue (if any) now synchronizes with the token or (if there is no job in the external queue) the token waits until it is requested by an arriving job. The synchronization of the jobs and the tokens ensures that the number of jobs in the inner system is never larger than the number of tokens. In the sequel, we also refer to the external queue as the *synchronization station*.

2.2. The solution procedure

We propose to solve the single-class OQN-PC with respect to the mean queue lengths by a procedure that consists of similar steps as in the method of Dallery [11]. However, we will show that the results of the new method are the same as those of the original aggregation method. Thus, we provide an *alternative view* of the aggregation method that is more in line with the idea of aggregation in CQNs (see Chandy et al. [9]).

The procedure is given by the following four steps.

1. Transform the OQN-PC into a CQN (consisting of all service stations and the synchronization station) in which the tokens are seen as the customers. Identify the synchronization station by the index $M + 1$. At the synchronization station the tokens may have to wait for an external resource (the jobs).

2. Replace the synchronization station by a load-dependent exponential server. For networks with exponential servers, the service rates of the replacing station are chosen such that the throughput $TH(N)$ of the resulting CQN, as measured at the replacing station, equals the external arrival rate λ of the jobs (see below). For networks with non-exponential servers, the service rates are chosen in a similar way that is consistent with the choice for networks with exponential servers.

3. Solve the resulting product-form CQN to find the mean number of tokens (and hence the mean number of jobs) at the service stations. To this aim, use an AMVA algorithm.

4. Find the mean number of jobs in the external queue by considering the external queue in isolation and by solving the birth and death process that describes its behavior.

Steps 2, 3 and 4 require some description in more detail in order for the procedure to be well-defined. First, notice that step 1 constitutes an exact transformation. No approximation is introduced by letting the tokens be the customers.

In step 2, we modify the CQN in a minimal way such that we are able to solve the resulting network. We set the load-dependent service rates $\nu_{M+1}(n)$ of the synchronization station equal to λ for $n > 1$: if there are tokens at the synchronization station then the first in line token has to wait for an exponential time with rate λ until it synchronizes with a job and leaves the station. We set $\nu_{M+1}(1)$ such that the throughput $TH(N)$ of the resulting CQN with N tokens is equal to the external arrival rate of the jobs λ. Thus, in that sense, the resulting CQN is consistent with the original OQN-PC.

We first describe the determination of $\nu_{M+1}(1)$ for networks with *exponential service stations*. Consider the CQN in which the synchronization station has been replaced by a load-dependent exponential server. This network has a product-form stationary distribution and therefore Norton's theorem (see Chandy et al. [10]) can be applied. In this case, Norton's theorem states that the behavior of the synchronization station is described by a birth and death process with states $n = 0, 1, \ldots, N$, (denoting the number of tokens at the station). The downcrossing rates are equal to the service rates $\nu_{M+1}(n)$ and the upcrossing rates are given by the throughput values $TH_{C_{M+1}}(N-n)$ of the complement network of the synchronization station, C_{M+1}. (The *complement network* C_m of station m is obtained by immediately redirecting tokens that arrive at station m according to the probabilities $P_{m\widetilde{m}}$, $\widetilde{m} \neq m$. Thus, the complement network of station m consists of all stations except station m. The new route that redirects the jobs is also called the *shortcut*.) Therefore, for the marginal probabilities $p_{M+1}(0)$ and $p_{M+1}(1)$ that there are 0 or 1 token(s) at the synchronization station, respectively, it must hold that

$$\frac{p_{M+1}(0)}{p_{M+1}(1)} = \frac{\nu_{M+1}(1)}{TH_{C_{M+1}}(N)}. \tag{1}$$

Furthermore, the number of jobs that arrive at the synchronziation station per time unit must be equal to the number of jobs (and tokens) that depart from the synchronization station per time unit:

$$\lambda = p_{M+1}(1)\nu_{M+1}(1) + \big[1 - p_{M+1}(0) - p_{M+1}(1)\big]\lambda, \tag{2}$$

which can be rewritten as

$$\nu_{M+1}(1) = \lambda\left(1 + \frac{p_{M+1}(0)}{p_{M+1}(1)}\right). \tag{3}$$

Combining equations (1) and (3) yields

$$\nu_{M+1}(1) = \frac{1}{q}\lambda, \tag{4}$$

where q is given by

$$q = 1 - \frac{\lambda}{TH_{C_{M+1}}(N)}. \tag{5}$$

In the method of Dallery [11] (for which the first two steps are identical to the first two steps of our method), the same choice is made for the service rates $\nu_{M+1}(n)$ of the load-dependent server. However, in [11], the choice for $\nu_{M+1}(1)$ has been based on different arguments, from which it does not follow that precisely this choice leads to the appropriate throughput.

Notice that for an ergodic network, it holds that $\lambda < TH_{C_{M+1}}(N)$ and therefore $0 < q < 1$. Thus, the replacing station serves at rate λ if there are $n > 1$ tokens present, and serves faster if there is only one token present.

For a network with only exponential service stations, the throughput $TH_{C_{M+1}}(N)$, and hence also the q, can be determined exactly, e.g., by a MVA algorithm. If there are service stations with *non-exponential* servers, then we propose to *approximate* the throughput $TH_{C_{M+1}}(N)$ by an AMVA algorithm and still to set the service rate $\nu_{M+1}(1)$ of the synchronization station according to equations (4) and (5), and set $\nu_{M+1}(n) = \lambda$ for $n > 1$. In other words, we pretend that for this non-product-form network Norton's theorem still holds. So, in this case two extra approximations are involved.

In step 3, we analyze a CQN, which consists of at least one load-dependent exponential server, and which may further consist of exponential and non-exponential servers. We propose to solve such networks with an AMVA algorithm, that at least yields exact results for product-form networks. The algorithm we use for this purpose combines the algorithm described by Buzacott and Shanthikumar [8, pp. 399–400] and certain approximations for the $M|G|c$ queue. These algorithms are recursive in the spirit of the exact MVA algorithm as described by Reiser and Lavenberg [16]. All AMVA algorithms used for the procedures described in this paper are given in the appendices. For the details of the algorithm used here, see appendix A.2.

In step 4, we analyze the synchronization station in isolation in order to find the mean number of jobs present there. This is done by extracting the state $n = 0$ of the Markov process from which we have obtained the load-dependent service rate $\nu_{M+1}(1)$. Consider this new Markov process with the states (n_t, n_j), where n_t denotes the number of tokens and n_j denotes the number of jobs at the synchronization station. The arrival rate of the jobs is λ and, according to Norton's theorem, the arrival rate of the tokens is $TH_{C_{M+1}}(N - n_t)$. (In conformance with step 3, we pretend that this is also true if there are non-exponential servers.) Since there can never be jobs and tokens at the synchronzation station at the same time (they would have been synchronized), the only feasible states are $(n_t, 0)$ for $n_t = 0, \ldots, N$, and $(0, n_j)$, for $n_j \geqslant 0$. The Markov process is thus a birth and death process and can be solved by solving the local balance equations (6) and (7):

$$\lambda p(n_t, 0) = TH_{C_{M+1}}(N - n_t + 1)p(n_t - 1, 0), \quad \text{for } n_t = 1, \ldots, N, \tag{6}$$

$$\lambda p(0, n_j) = TH_{C_{M+1}}(N)p(0, n_j + 1), \quad \text{for } n_j \geqslant 0. \tag{7}$$

From equations (6) and (7) we find the probabilies $p(0, n_j)$, for $n_j > 0$, and from them we find the mean number of jobs in the external queue EL_{ext}:

$$EL_{\text{ext}} = \sum_{n_j=1}^{\infty} n_j p(0, n_j) = p(0,0)\,\frac{\rho}{(1-\rho)^2},$$

where $\rho = \lambda / TH_{C_{M+1}}(N)$ and $p(0,0)$ is given by

$$p(0,0) = \prod_{k=1}^{N} \frac{\lambda}{TH_{C_{M+1}}(k)}\, p(N,0)$$

with

$$p(N,0) = \left[\sum_{n_t=0}^{N} \prod_{k=1}^{N-n_t} \frac{\lambda}{TH_{C_{M+1}}(k)} + \prod_{k=1}^{N} \frac{\lambda}{TH_{C_{M+1}}(k)} \frac{\rho}{1-\rho} \right]^{-1}.$$

Summarizing, for networks with only exponential servers, the approximations made in the procedure described above concern the replacement of the synchronziation station by a load-dependent exponential server and the analysis of the synchronization station in isolation. For networks with non-exponential servers the extra approximations are that we pretend that Norton's theorem still holds and that we approximate the throughputs of both the complement network for all $n \leqslant N$, and the resulting network using an AMVA algorithm.

2.3. Relations with other methods

As mentioned in section 1, there are two known methods to approximately evaluate the performance of the OQN-PC: the aggregation method and the method of Dallery. In this section, we show how these two methods and the new method, described here, relate.

Consider first the *aggregation method*, conform the description given by Buzacott and Shanthikumar [7]. This method proceeds as follows. First, a CQN is constructed from the inner system by redirecting departing jobs to a station in the inner system according to the routing probabilities $P_{m0} \cdot P_{0\widetilde{m}}$ (where P_{m0} denotes the probability that a part leaves the system after its visit to station m and $P_{0\widetilde{m}}$ is the probability that the first operation of a part is at station $\widetilde{m}$, respectively). Second, for the populations $n = 1, \ldots, N$, the throughput $TH_{\text{aggr}}(n)$ of this CQN is computed. Third, the OQN-PC is replaced by an $M|M(n)|1$ queue, with load-dependent service rates $\mu(n) = TH_{\text{aggr}}(\min(n, N))$. Finally, the birth and death process describing the behavior of this queue is solved to obtain the marginal probabilities $p_{\text{aggr}}(n)$, and from them the mean total number of jobs at the OQN-PC is computed.

It is easily verified that the CQN constructed by the aggregation method is identical to the complement network of the synchronization station in the new method. It then also follows that the birth and death process solved in the aggregation method is

the same as the birth and death process describing the behavior of the synchronization station in isolation in the new method. In other words, *the aggregation method and the new method yield the same value for the mean number of jobs in the external queue and for the mean total queue length*. The new method additionally gives the mean queue length at each service station.

Consider next the *method of Dallery*. Here, the non-exponential servers are assumed to have service times with a Coxian-2 distribution. The method first transforms the OQN-PC into a CQN in the same way as described in subsection 2.2 (we adopted this transformation from Dallery). It then replaces the synchronization station *and all non-exponontial servers* by load-dependent exponential servers, conform the method of Marie [13]. This is implemented by the following iterative procedure that stops upon the convergence of the load-dependent service rates. First, the load-dependent service rates are set equal to an initial value. Next, a number of iterations is performed, where each iteration consists of two steps. In the first step of iteration i the resulting product-form CQN is analyzed to obtain the state-dependent arrival rates:

$$\lambda_m^{[i]}(n) = TH_{C_m}^{[i]}(N-n), \tag{8}$$

where $\lambda_m^{[i]}(n)$ denotes the state-dependent arrival rate at station m in the network of iteration i if there are n tokens at station m, and $TH_{C_m}^{[i]}(N-n)$ denotes the throughput of the complement network of station m in the network of iteration i, with population $N-n$. In the second step the queues are analyzed in isolation to obtain the new values of the load-dependent service rates, $\mu_m^{[i+1]}(n)$. The non-exponential servers are analyzed as $\lambda(n)|C_2|1|N$ queues. The synchronization station is analyzed conform the birth and death process as given in subsection 2.2, with the throughput of the original complement of the synchronization station $TH_{C_{M+1}}(N-n)$ replaced by the current values of $\lambda_{M+1}^{[i]}(n)$. These analyses yield the marginal probabilities $p_m(n)$ that there are n tokens at station m. Finally, the new values of the load-dependent service rates are found from the following equation:

$$\mu_m^{[i+1]}(n) = \lambda_m^{[i]}(n-1)\,\frac{p_m(n-1)}{p_m(n)}. \tag{9}$$

After convergence of the service rates, the mean number of tokens at the service stations and the mean number of jobs at the synchronization station are computed using the values of the marginal probabilities found in the last iteration.

In principle, the method of Dallery does not require that the service time have a Cox-2 distribution. For the analysis in isolation of a queue to be performed using Markov process properties, it is required, however, that the service times have a phase-type distribution. Therefore, networks with deterministic service times or close to deterministic service times cannot be easily solved, unless a different analysis in isolation can be provided. In our method the deterministic service times are taken care of automatically in the AMVA algorithm.

For OQNs-PC with only exponential servers, the method of Dallery only needs to find the replacing load-dependent service rates for the synchronization station. Since

the complement does not change, the state-dependent arrival rates at the synchronization station do not change and no iterations are required. This means that the analysis of the synchronization station in isolation from which the eventual results are obtained concerns the same birth and death process as that of our AMVA-based method and the aggregation method. Thus, in that case, *the three methods are equivalent*.

If there are non-exponential servers, then it will in general no longer hold that the eventual state-dependent arrival rates at the synchronization station as obtained by the method of Dallery equal the throughput values of the original complement of the synchronization station. In particular, the state-dependent service rates will account for the service rates at the other stations and thus also for the external arrival rate. In that case, the method of Dallery yields results that differ from the results generated by the aggregation method or the new method.

3. The multiclass OQN-PC: shared tokens

In this section, we consider the OQN-PC with multiple job classes that have a common population constraint. The common population constraint is represented by one pool of tokens which is shared by the different classes. This queueing network models an FMS which manufactures multiple part types and which uses general purpose pallets for the different part types.

The model for this case is described as follows. Denote by R the number of classes (numbered from 1 to R) and, as before, denote by M the number of machines (numbered from 1 to M) and by N the number of tokens. As in the single-class network, there is *one* synchronization station, where each job synchronizes with a token. The synchronization station is identified by the index $M+1$. Class r jobs arrive at the system according to a Poisson process with rate $\lambda^{(r)}$, $r = 1, \ldots, R$. Class r jobs have a Markovian routing that is characterized by the routing matrix $P^{(r)}$, $r = 1, \ldots, R$. The service time at station m for class r (if class r visits station m) has a general distribution with known first two moments $ES_m^{(r)}$ and $E(S_m^{(r)})^2$. At each station the jobs are served in a first-come first-served order.

Denote by $n_I^{(r)}$ the number of class r jobs in the inner system. The shared tokens now arrange that the total number of jobs in the inner system is smaller than or equal to the total number of tokens:

$$\sum_{r=1}^{R} n_I^{(r)} \leqslant N.$$

This is also referred to as the common population constraint for the different classes.

3.1. The procedure

We want to generalize the procedure as described in subsection 2.2. The major modification concerns step 1 of that procedure. In step 1, the OQN-PC with multiple

job classes and shared tokens is now transformed into a single chain CQN with multiple classes in which tokens change class at the synchronization station. We describe this transformation in more detail before we present the procedure for this case.

Denote by $\lambda = \sum_{r=1}^{R} \lambda^{(r)}$ the total external arrival rate of the jobs, by $\lambda_m^{(r)}$ the arrival rate of class r jobs at station m and by $\lambda_m = \sum_{r=1}^{R} \lambda_m^{(r)}$ the total arrival rate of jobs at station m, all in the OQN-PC. For each class r, the rates $\lambda_m^{(r)}$ follow directly from the routing matrix $P^{(r)}$. Observe that with probability $\lambda^{(r)}/\lambda$ an arriving job is of class r and, with the same probability, the first job at the synchronization station is of class r. Therefore, in the resulting single chain CQN, a token leaves the synchronization station as a class r token with probability $\lambda^{(r)}/\lambda$. This, together with the original routing matrices $P^{(r)}$, identifies the routing matrix $\widetilde{P}$ of the single chain tokens. Element $\widetilde{P}_{mn}^{rs}$ of $\widetilde{P}$ denotes the probability that a token of class r upon a service completion at station m moves to station n and there becomes a class s token; notice that the dimension of $\widetilde{P}$ is $R(M+1)$, while the dimension of $P^{(r)}$ is $M+1$. We find

$$
\begin{aligned}
&\widetilde{P}_{mn}^{rr} = P_{mn}^{(r)}, && m,n = 1,\ldots,M,\\
&\widetilde{P}_{mn}^{rs} = 0, && m,n = 1,\ldots,M,\ r \neq s,\\
&\widetilde{P}_{mM+1}^{rr} = P_{m0}^{(r)}, && m = 1,\ldots,M,\\
&\widetilde{P}_{M+1m}^{rs} = \frac{\lambda^{(s)}}{\lambda} P_{0m}^{(s)}, && m = 1,\ldots,M,\\
&\widetilde{P}_{M+1,M+1}^{rs} = 0.
\end{aligned}
$$

From the routing matrix $\widetilde{P}$ the visit ratios $\widetilde{V}_m^{(r)}$ in the CQN can be computed up to a normalization constant. The visit ratios can also be easily found from the visit ratios for the original OQN-PC. Denote by $V_m^{(r)}$ the visit ratio of class r jobs to station m in the OQN-PC. Then from the routing matrix $P^{(r)}$ we find that

$$
V_m^{(r)} = \frac{\lambda_m^{(r)}}{\lambda^{(r)}}, \quad m = 1,\ldots,M+1.
$$

If we now choose to normalize the visit ratios for the CQN such that

$$
\widetilde{V}_{M+1} = \sum_{r=1}^{R} \widetilde{V}_{M+1}^{(r)} = 1,
$$

we find that

$$
\begin{aligned}
&\widetilde{V}_{M+1}^{(r)} = \frac{\lambda^{(r)}}{\lambda}, \quad r = 1,\ldots,R, \quad \text{and}\\
&\widetilde{V}_m^{(r)} = V_m^{(r)} \cdot \widetilde{V}_{M+1}^{(r)} = \frac{\lambda_m^{(r)}}{\lambda^{(r)}} \frac{\lambda^{(r)}}{\lambda} = \frac{\lambda_m^{(r)}}{\lambda}, \quad r = 1,\ldots,R,\ m = 1,\ldots,M.
\end{aligned}
$$

Define the visit ratio $\widetilde{V}_m$ of an arbitrary token to station m as the mean number of visits to station m between two visits to station $M+1$. It holds that

$$\widetilde{V}_m = \sum_{r=1}^{R} \widetilde{V}_m^{(r)} = \sum_{r=1}^{R} \frac{\lambda_m^{(r)}}{\lambda} = \frac{\lambda_m}{\lambda}, \quad m = 1, \ldots, M+1.$$

Given the visit ratios of an arbitrary token, we can formulate an AMVA algorithm for a single chain CQN with different classes. This algorithm performs the same number of iterations as the AMVA algorithm for single-class networks. The algorithm for this case, however, requires more variables and more iteration steps per iteration. For example, there is a variable for the lead time of a class r job at each station m given that there is a total number of n tokens in the network. The resulting AMVA algorithm is given in appendix A.3.

The resulting procedure for this case is given by the following five steps:

1. Transform the OQN-PC into a single chain multiple class CQN (that includes the synchronization station).
2. Replace the synchronization station by a load-dependent exponential server. The service rates only depend on the total number of tokens present. The service rates of the replacing station are chosen such that the throughput $TH(N)$ of the CQN is equal to the *total* external arrival rate λ of the jobs.
3. Approximately solve the resulting CQN to find the mean number of jobs per class at the service stations. To this aim, use an AMVA algorithm for CQNs with a single chain (the aggregate class) and multiple classes (see appendix A.3). This yields *the mean number of jobs per class* at each service station.
4. Compute the mean total number of jobs in the external queue (EL_{ext}) by considering the external queue in isolation and solving the birth and death process that describes its behavior.
5. Compute the mean number of class r jobs in the external queue:

$$EL_{\text{ext}}^{(r)} = \frac{\lambda^{(r)}}{\lambda} EL_{\text{ext}}.$$

In step 2, we set the load-dependent service rate $\nu_{M+1}(n)$ equal to λ, for $n > 1$, and $\nu_{M+1}(1)$ in a similar way to equations (4) and (5):

$$\nu_{M+1}(1) = \frac{1}{q}\lambda, \tag{10}$$

where q is given by

$$q = 1 - \frac{\lambda}{TH_{C_{M+1}}(N)}, \tag{11}$$

and $TH_{C_{M+1}}(N)$ denotes the total throughput of the complement network of station $M+1$ with N tokens.

Remark. For OQNs-PC with multiple classes and shared tokens and only exponential servers *with the same service rate for different classes*, the procedure can be somewhat simplified. In that case, the same results are obtained by solving a single chain CQN without accounting for the different classes in the equations of the MVA algorithm. The internal queue lengths per class are then obtained by computing them in the end according to

$$EL_m^{(r)} = \frac{\lambda_m^{(r)}}{\lambda_m} EL_m.$$

3.2. Relation with other methods

The same transformation as presented here (which transforms a network with multiple classes into a single chain network with multiple classes) can be applied for both the aggregation method and the method of Dallery. In [6], we did this for the aggregation method. The generalization for the method of Dallery is reported in [4]. These generalizations of the three methods relate to each other in the same way as the methods for the single-class networks.

Although the generalization suggested here is a logical one, it is by no means the only one to generalize the methods for this situation. In [6], we have presented two other ways to generalize the aggregation method for networks with multiple classes and shared tokens.

4. The multiclass OQN-PC: dedicated tokens

In this section, we consider the OQN-PC with multiple job classes with dedicated population constraints represented by a set of dedicated tokens per class. This models an FMS which manufactures multiple part types for which the pallets are dedicated to the part type, for example due to technical conditions.

The model for this case is depicted in figure 2. Denote again by M the number of machines (which are numbered from 1 to M) and by R the number of job classes (which are numbered from 1 to R). For class r there are $N^{(r)}$ dedicated tokens available. Denote by $\vec{N} = (N^{(1)}, \ldots, N^{(R)})$ the token population. There are now R synchronization stations; one for each class. The synchronization station for class r is identified by the index $M + r$. Class r jobs arrive at their synchronization station according to a Poisson process with rate $\lambda^{(r)}$. There they synchronize with a token of their class. Class r jobs have a Markovian routing that is characterized by the routing matrix $P^{(r)}$. Upon the external departure of a class r job, the class r token moves to synchronization station $M + r$. The service time at station m for class r (if class r visits station m) has a general distribution with known first two moments $ES_m^{(r)}$ and $E(S_m^{(r)})^2$. At each station the jobs are served in a first-come first-served order.

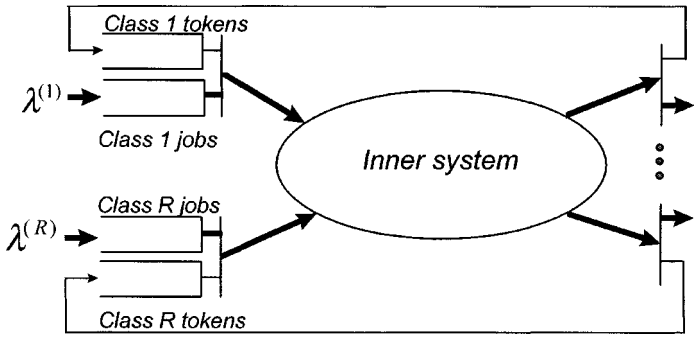

Figure 2. The multiclass OQN-PC with dedicated tokens.

Denote by $n_I^{(r)}$ the number of class r jobs in the inner system. The dedicated tokens now arrange that the number of class r jobs in the inner system is smaller than or equal to the number of class r tokens:

$$n_I^{(r)} \leqslant N^{(r)}, \quad r = 1, \ldots, R. \tag{12}$$

Equation (12) represents the population constraint for each class $r = 1, \ldots, R$.

4.1. The procedure

For this model, two observations should be made. First, notice that each of the synchronization stations is visited by just one job class; station $M + r$ is visited by class r only. This allows us to replace the synchronization stations by single-class load-dependent exponential servers, where the load is given by the number of class r jobs, and thus to generalize the analysis in isolation for synchronization stations in a straightforward way. Second, in a network with $R > 1$ classes, the complement network of a synchronization station contains $R-1 > 0$ other synchronization stations. In this case, we want to set the load-dependent service rates of station $M+r$ according to a straightforward generalization of equations (4) and (5). We define $\nu_{M+r}(n) = \lambda^{(r)}$, for $n > 1$, and

$$\nu_{M+r}(1) = \frac{1}{q^{(r)}}\,\lambda^{(r)}, \tag{13}$$

where $q^{(r)}$ is given by

$$q^{(r)} = 1 - \frac{\lambda^{(r)}}{TH^{(r)}_{C_{M+r}}(\vec{N})}, \tag{14}$$

and $TH^{(r)}_{C_{M+r}}(\vec{N})$ denotes the throughput of class r in the complement network of station $M + r$, as measured at the shortcut. The throughput $TH^{(r)}_{C_{M+r}}(\vec{N})$ depends on the load-dependent service rates $\nu_{M+s}(n)$, $s \neq r$. Thus the service rates of the different synchronization stations depend on each other and an iterative procedure

is required to find them. Since the procedure uses an AMVA algorithm it yields approximations for the throughput values. Observe however that we can still guarantee that $\lambda^{(r)} = TH_{M+r}(\vec{N})$ (a similar analysis as given for single-class networks can be done here).

The procedure for this case is given by the following four steps:

1. Transform the OQN-PC into a multiclass CQN (including the R synchronization stations).
2. Replace the synchronziation stations by load-dependent exponential servers:
   (a) Set the load-dependent service rates for stations $M+1$ to $M+R$ to some initial value. For example, set $v_{M+r}(1) = \lambda^{(r)}$; this corresponds to rejecting jobs that upon their arrival do not find a token of their class in the external queue.
   (b) For $r = 1, \ldots, R$, solve the complement CQN of station $M+r$, C_{M+r}, by an AMVA algorithm. This yields the class r throughput of the complement CQN of station $M+r$, with population $\vec{N}$, $TH^{(r)}_{C_{M+r}}(\vec{N})$.
   (c) Reset the load-dependent service rates according to equations (13) and (14).
   (d) Repeat step 2(b) and step 2(c) until convergence of the service rates.
3. Solve the resulting CQN with an AMVA algorithm.
4. Compute the number of jobs in each synchronization station by solving the birth and death process that describes its behavior.

Observe that the procedure for this case is again very similar to the procedure for the single-class case. The extra difficulty arises mainly in step 2. To replace the synchronization station by load-dependent exponential servers we need to perform an iterative (sub)procedure. This of course greatly affects the efficiency of the procedure.

The AMVA algorithm to be applied in step 3 is formulated in appendix A.4. The algorithm needed in step 2(b) is similar to the algorithm in appendix A.4. Only some minor modifications are required.

The computation performed in step 4 is basically identical to that of the last step in the procedure for the single-class case. This is due to the fact that each synchronziation station is only visited by one class.

4.2. Relation with other methods

The method described in this section follows the lines of the generalization of the method of Dallery by Perros et al. [15]. Perros et al. demand that the inner system is of the BCMP-type. Therefore, after the synchronization stations have been replaced by load-dependent exponential servers, the network has a product-form solution. This allows them to use any exact method to compute the state-dependent arrival rates at the stations. If there are only exponential servers with the same rate for different classes

(we assumed first-come first-served service discipline at all stations), then the method of Perros et al. is in fact equivalent to the new method.

If there are exponential servers with *different service rates for different classes*, then a further straightforward generalization of the method of Dallery requires the analysis in isolation of a complex queue. The behavior of such an exponential server with different rates for different classes is described by a multidimensional random walk, which cannot be solved efficiently. Therefore, Baynat and Dallery [2] propose to introduce an extra approximation to avoid these computational complexities. They decompose a network with R classes into R single-class networks, each of which is associated to one class. When analyzing a station visited by multiple classes in isolation, the other classes are accounted for when estimating the load-dependent service rates.

Our new method generalizes the method of Perros et al. by not modifying the general servers and applying an AMVA algorithm to the complement networks of the synchronization stations. Our AMVA algorithm can easily deal with single-class load-dependent servers and multiclass general servers.

To compare the methods extensive numerical experiments are required. We give a preliminary comparison based on logic. The methods can be compared with respect to three criteria: accuracy, generality, and complexity. The accuracy of a method indicates how accurate the results of the method are. The generality indicates how general the networks to which the method can be applied may be. The complexity indicates which size of problems the method can deal with. In table 1, we summarize the comparison of the method of Perros, Dallery and Pujolle (PDP), the method of Baynat and Dallery (BD) and the new method (MVA) for this case.

The MVA-based method can deal with general networks (the MVA can even be extended in order to deal with more complicated servers or subsystems). In general, the approximations may be expected to be more accurate than those of the method of Baynat and Dallery (because they apply an extra approximate decomposition step), but the size of the problem which it is able to solve is smaller.

Other generalizations of the aggregation method have also been proposed, see, for example, [12,20]. However, these have no direct relation with the method presented here.

Table 1
Three methods for the multiclass OQN-PC with dedicated tokens compared.

	Accuracy	Generality	Complexity
PDP	+	−	−
BD	−	+	+
MVA	+	+	−

5. Further extensions and numerical experiments

The procedures proposed in this paper can be extended to deal with multiclass networks in which tokens are dedicated to subsets of the classes (where the subsets form a partition of the set of classes). In that case the classes in a subset share one pool of tokens. Such a network can be "reduced" to a multichain network in which jobs may change classes and with dedicated pallets per chain. The reduction proceeds along the lines as described in section 3 for shared tokens.

It is desirable to compare the different methods for the different cases by extensive numerical experiments. As an illustration of the (potential) accuracy of our new method we present the results of some numerical experiments (for more numerical experiments, see [5]). We present results for the case with single classes as a base line and we present results for the multiclass case with dedicated tokens, because that case constitutes the most important extension of the single-class case. Results for the multiclass case with shared tokens may be expected to be of better accuracy than results for the case with dedicated tokens.

Our experiments are for networks with two single-server stations both with visit ratio equal to one. These visit ratios are realized by two types of networks: the pure flow shop and the pure job shop. Because our method is insensitive to routing probabilities and only depends on the routing through the visit ratios, the results of the approximation for these two types of networks are identical. The routing of the two shop types is characterized by the matrices P_F and P_J,

$$P_F = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad P_J = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix},$$

in which the first row and column are numbered 0 and represent the external arrivals and departures and the second row (and column) and the third row (and column) represent station 1 and station 2, respectively. For all instances, we present the mean number of jobs in the external queue EL_{ext}, the mean number of jobs in the inner system EL_{int} and the mean number of jobs in the system EL_{tot}, as obtained by the new method and as obtained by simulation for the two shop types. In the single-class instances, the service times at both stations have an exponential distribution with rate $\mu = 1$ and the jobs arrive according to a Poisson arrival process with rate $\lambda = 0.6$. The number of tokens varies from 2 (the mimimum required to keep the system ergodic) to 10. The results for these instances are given in table 2. We present two sets of instances for the multiclass case with dedicated tokens. The first set concerns symmetrical instances. The second concerns asymmetrical instances. For both sets there are two classes. The classes in the symmetrical instances have equal arrival rates $\lambda = 0.3$, equal service times at the different stations which have an exponential distribution with rate $\mu = 1$, and an equal number of tokens, which varies over the instances. The results for the symmetrical instances are given in table 3. The asymmetrical instances are characterized by again equal arrival rates for the two classes ($\lambda = 0.3$), and an

Table 2
The mean queue lengths for the single-class case as obtained by simulation for both shop types and by the solution procedure of subsection 2.2.

N	Flow shop			Job shop			New method		
	L_{ext}	L_{int}	L_{tot}	L_{ext}	L_{int}	L_{tot}	L_{ext}	L_{int}	L_{tot}
2	5.807	1.748	7.555	7.701	1.753	9.454	7.477	1.754	9.231
3	1.785	2.127	3.912	2.371	2.144	4.514	2.274	2.147	4.421
4	0.942	2.401	3.343	1.227	2.421	3.649	1.154	2.422	3.577
5	0.549	2.592	3.142	0.706	2.609	3.315	0.663	2.613	3.276
8	0.143	2.889	3.032	0.170	2.892	3.062	0.154	2.889	3.043
10	0.057	2.955	3.013	0.067	2.956	3.023	0.061	2.953	3.014
∞	0	3	3	0	3	3	0	3	3

Table 3
The mean queue lengths for the multiclass symmetrical case as obtained by simulation for both shop types and by the solution procedure of subsection 4.1.

$(N^{(1)}, N^{(2)})$	Flow shop			Job shop			New method		
	$L_{\text{ext}}^{(r)}$	$L_{\text{int}}^{(r)}$	$L_{\text{tot}}^{(r)}$	$L_{\text{ext}}^{(r)}$	$L_{\text{int}}^{(r)}$	$L_{\text{tot}}^{(r)}$	$L_{\text{ext}}^{(r)}$	$L_{\text{int}}^{(r)}$	$L_{\text{tot}}^{(r)}$
(1,1)	3.682	0.847	4.529	5.101	0.847	5.948	4.241	0.835	5.076
(2,2)	0.642	1.135	1.777	0.855	1.146	2.001	0.697	1.123	1.820
(3,3)	0.257	1.308	1.565	0.340	1.316	1.656	0.274	1.297	1.571
(4,4)	0.120	1.403	1.523	0.150	1.409	1.559	0.122	1.394	1.516
∞	0	1.5	1.5	0	1.5	1.5	0	1.5	1.5

equal number of tokens for the two classes ($N^{(1)} = N^{(2)} = 2$). The classes in the asymmetrical instances differ through their service times. The service times are chosen such that the utilization of the stations $\rho = \lambda(ES_m^{(1)} + ES_m^{(2)}) = 0.6$, $m = 1, 2$. The results for the asymmetrical instances are given in table 4.

The results obtained from simulation concern the mean values of the 95%-confidence intervals which where obtained by simulating until the half-width of the interval is smaller than 1% of its mean.

The most important observations made from the results are:

1. In general, the approximations are good for the single-class case, especially for the job shop.
2. For the multiclass case, the results are better if the inner system is closer to a product-form type network. In particular, if the ratio between the mean service time of class 1 and the mean service time of class 2 at one station is close to one, then the approximation is reasonably accurate.
3. The approximation is more accurate for the single-class case than for the multiclass case.
4. The accuracy of the approximation decreases when the utilization of the tokens increases. In our experiments this is recognized by comparing two instances with

Table 4
The mean queue lengths for the multiclass asymmetrical case as obtained by simulation for both shop types and by the solution procedure of subsection 4.1.

$ES^{(r)}$	r	Flow shop			Job shop			New method		
		$L_{\text{ext}}^{(r)}$	$L_{\text{int}}^{(r)}$	$L_{\text{tot}}^{(r)}$	$L_{\text{ext}}^{(r)}$	$L_{\text{int}}^{(r)}$	$L_{\text{tot}}^{(r)}$	$L_{\text{ext}}^{(r)}$	$L_{\text{int}}^{(r)}$	$L_{\text{tot}}^{(r)}$
(0.5,0.5)	1	0.806	1.111	1.917	0.795	1.055	1.850	0.420	1.015	1.435
(1.5,1.5)	2	1.405	1.385	2.790	1.956	1.404	3.360	1.682	1.374	3.056
(0.75,0.75)	1	0.606	1.088	1.695	0.731	1.074	1.805	0.515	1.051	1.566
(1.25,1.25)	2	0.838	1.236	2.074	1.198	1.258	2.455	1.033	1.231	2.264
(0.9,0.9)	1	0.593	1.112	1.705	0.793	1.112	1.905	0.611	1.089	1.700
(1.1,1.1)	2	0.698	1.175	1.873	0.966	1.186	2.152	0.807	1.161	1.968
(0.5,1.5)	1	0.912	1.236	2.148	1.104	1.211	2.315	0.847	1.198	2.045
(1.5,0.5)	2	0.838	1.190	2.028	1.125	1.212	2.337	0.847	1.198	2.045
(0.75,1.25)	1	0.711	1.170	1.881	0.914	1.162	2.075	0.732	1.142	1.874
(1.25,0.75)	2	0.676	1.146	1.823	0.928	1.162	2.089	0.732	1.142	1.874
(0.9,1.1)	1	0.656	1.136	1.799	0.847	1.144	1.990	0.702	1.126	1.828
(1.1,0.9)	2	0.653	1.139	1.792	0.859	1.145	2.004	0.702	1.126	1.828

equal arrival rates and different numbers of tokens. Another way would be to compare two instances with equal numbers of tokens and different arrival rates.

Appendix A: MVA algorithms

In this appendix we present four algorithms: (i) an exact MVA algorithm for single-class networks with exponential, load-dependent servers; (ii) an AMVA algorithm that can be used in the solution procedure for single-class OQNs-PC; (iii) a single chain multiclass AMVA algorithm that can be used in the solution procedure for multiclass OQNs-PC with shared tokens; (iv) a multiclass AMVA algorithm that can be used in the solution procedure for multiclass OQNs-PC with dedicated tokens.

The first algorithm is copied from the textbook of Buzacott and Shanthikumar [8]. The second AMVA algorithm is an algorithm for a network consisting of one exponential, load-dependent service station and M service stations with possibly multiple servers and generally distributed service times. It is based on an algorithm formulated in [8] for a network with only the latter type of service stations. Our algorithm is adapted for the presence of the exponential, load-dependent service station. Further, we use a modified expression for the lead times at the general service stations; for the details on this modification, see appendix A.2. Both the third and the fourth algorithm may be seen as a generalization of the second algorithm. The difference between them and corresponding algorithms in [8] is similar as for the second algorithm.

We like to stress that the AMVA algorithms presented here are by no means the only possible algorithms that one may use. They just serve as an illustration of what is possible.

A.1. Single-class exact MVA

An algorithm for single-class product-form CQNs with $M+1$ load-dependent exponential service stations (with rates $\mu_m(k)$, $k \geqslant 0$, for station m) is given on pages 373–374 in [8] and reads as follows:

1. Set $p_m(0|0) = 1,\ m = 1, \ldots, M+1$.
2. For $n = 1, \ldots, N$, compute:
   (a) for $m = 1, \ldots, M+1$,

$$ET_m(n) = \sum_{k=0}^{n-1} \frac{k+1}{\mu_m(k+1)}\, p_m(k|n-1); \tag{15}$$

   (b)
$$TH(n) = \frac{n}{\sum_{m=1}^{M+1} \nu_m\ ET_m(n)}; \tag{16}$$

   (c) for $m = 1, \ldots, M+1$,

$$p_m(k|n) = \frac{\nu_m TH(n)}{\mu_m(k)}\, p_m(k-1|n-1), \quad k = 1, \ldots, n, \tag{17}$$

$$p_m(0|n) = 1 - \sum_{k=1}^{n} p_m(k|n). \tag{18}$$

A.2. Single-class AMVA

AMVA algorithms for the general single-class CQN are given on pages 378–380 of [8]. We suggest a slightly different algorithm for a CQN with $M+1$ stations, which are as follows. Station $M+1$ is a load-dependent exponential server with rates

$$\mu(1) = \frac{\lambda}{q} \quad \text{and } q = 1 - \frac{\lambda}{TH_{C_{M+1}}(N)},$$
$$\mu(n) = \lambda, \quad \text{for } n > 1.$$

Further, station $m = 1, \ldots, M$ is a general service station with c_m parallel servers and with service times S_m, with known first two moments ES_m and ES_m^2.

We first rewrite equation (15) for the lead time at station $M+1$:

$$\begin{aligned} ET_{M+1}(n) &= \sum_{k=1}^{n-1} \frac{k+1}{\lambda}\, p_{M+1}(k|n-1) + p_{M+1}(0|n-1)q\, \frac{1}{\lambda} \\ &= \big[EL_{M+1}(n-1) + Q_{M+1}(n-1) + p_{M+1}(0|n-1)q\big]\frac{1}{\lambda}, \end{aligned} \tag{19}$$

where

$$EL_{M+1}(n) = \sum_{k=1}^{n} k p_{M+1}(k|n) \quad \text{and} \quad Q_{M+1}(n) = \sum_{k=1}^{n} p_{M+1}(k|n).$$

For the other stations we define the mean number of jobs in the queue (excluding jobs in service) $E\widetilde{L}_m(n)$ and the probability that all servers are busy $Q_m(n)$ as follows (both in a CQN with n tokens):

$$E\widetilde{L}_m(n) = \sum_{k=c_m}^{n} (k - c_m) p_m(k|n),$$

$$Q_m(n) = \sum_{k=c_m}^{n} p_m(k|n).$$

Furthermore, we denote by $ES_{\text{rem},m}$ the mean time until the first departure at station m. The lead time $ET_m(n)$ of a job in a network with n jobs consists of the waiting time until the first departure if all servers are busy ($Q_m(n-1)\ ES_{\text{rem},m}$), and of the waiting time for the jobs in the queue ($E\widetilde{L}_m(n-1)\ ES_m/c_m$) and of its own service time ES_m. Thus, we find

$$ET_m(n) = Q_m(n-1) ES_{\text{rem},m} + E\widetilde{L}_m(n-1) \frac{ES_m}{c_m} + ES_m. \tag{20}$$

We approximate $ES_{\text{rem},m}$ conform the time until the first departure for an $M|G|c_m$ queue, see Tijms [21, p. 346]. In fact, with this choice we deviate from [8]. Further, in our expression (20) for the lead time, we maintain the variable $Q_m(n-1)$ for the probability that all servers are busy, while in [8] for this probability an additional approximation is introduced.

Equation (20) can be shown to be exact for networks with only exponential servers. One could give a similar interpretation to equation (19), where $EL_{M+1}(n-1)/\lambda$ may be seen as the job's total waiting time, and $[Q_{M+1}(n-1)+p_{M+1}(0|n-1)q]/\lambda$ is the job's service time. However, this interpretation is mathematically incorrect.

We now obtain the following AMVA algorithm:

1. Initialize. Set $p_m(0|0) = 1$, $Q_m(0) = 0$, $E\widetilde{L}_m(0) = 0$, $m = 1, \ldots, M$ and $p_{M+1}(0|0) = 1$, $Q_{M+1}(0) = 0$, $EL_{M+1}(0) = 0$.

2. Preprocessing. For $m = 1, \ldots, M$,

$$ES_{\text{rem},m} = \frac{c_m - 1}{c_m + 1} \frac{ES_m}{c_m} + \frac{2}{c_m + 1} \frac{1}{c_m} \frac{ES_m^2}{2ES_m}.$$

   (a) for $m = 1, \ldots, M$,

$$ET_m(n) = Q_m(n-1)\ ES_{\text{rem},m} + E\widetilde{L}_m(n-1)\ \frac{ES_m}{c_m} + ES_m$$

and

$$ET_{M+1}(n) = \big[EL_{M+1}(n-1) + Q_{M+1}(n-1) + p_{M+1}(0|n-1)q\big]\frac{1}{\lambda};$$

(b) $$TH(n) = n/\sum_{m=1}^{M+1} \nu_m ET_m(n);$$

(c) for $m = 1, \ldots, M$, and for $l = 1, \ldots, \min(c_m - 1, n)$,

$$p_m(l|n) = \frac{ES_m}{l}\,\nu_m TH(n) p_m(l-1|n-1);$$

(d) for $m = 1, \ldots, M$, if $n < c_m$, $Q_m(n) = 0$, otherwise,

$$Q_m(n) = \frac{ES_m}{c_m}\,\nu_m TH(n)\big[Q_m(n-1) + p_m(c_m - 1|n-1)\big]$$

and

$$Q_{M+1}(n) = \frac{TH(n)}{\lambda}\big[Q_{M+1}(n-1) + p_{M+1}(0|n-1)q\big];$$

(e) for $m = 1, \ldots, M$

$$p_m(0|n) = 1 - \sum_{l=1}^{\min(c_m-1,n)} p_m(l|n) - Q_m(n)$$

and

$$p_{M+1}(0|n) = 1 - Q_{M+1}(n);$$

(f) for $m = 1, \ldots, M$, if $n < c_m$, $\widetilde{EL}_m(n) = 0$, otherwise,

$$\widetilde{EL}_m(n) = \frac{ES_m}{c_m}\,\nu_m TH(n)\big[\widetilde{EL}_m(n-1) + Q_m(n-1)\big];$$

(g) $$EL_{M+1}(n) = TH_{M+1}(n)\; ET_{M+1}(n).$$

A.3. Single chain multiclass AMVA

The notation used for this AMVA algorithm is similar to that of the previous algorithm. The extensions of the notation concern the superscripts (r) identifying the class. It is assumed that the visit ratios required for this AMVA algorithm are already computed in step 1 of the overall procedure. Furthermore, the variable q is defined as before (equations (10) and (11)). Note that for this, we need to analyze the complement of the synchronization station and therefore we need the routing matrix $\widehat{P}$ of this complement network:

$$\widehat{P}^{rs}_{mn} = P^{(r)}_{mn} \cdot 1_{\{r=s\}} + P^{(r)}_{m0}\frac{\lambda^{(s)}}{\lambda}P^{(s)}_{0n},$$

where the $P^{(r)}$ ($r = 1, \ldots, R$) are the routing matrices of the original OQN-PC.

1. Initialize. Set $p_m(0|0) = 1$, $Q_m(0) = 0$, $E\widetilde{L}_m(0) = 0$, $m = 1, \ldots, M$ and $p_{M+1}(0|0) = 1$, $Q_{M+1}(0) = 0$, $EL_{M+1}(0) = 0$.
2. For $n = 1, \ldots, N$, compute:
   (a) for $r = 1, \ldots, R$, and $m = 1, \ldots, M$,

$$ET_m^{(r)}(n) = Q_m(n-1)ES_{\text{rem},m}(n) + E\widetilde{L}_m(n-1)\frac{ES_m(n-1)}{c_m} + ES_m^{(r)},$$

   where

$$ES_{\text{rem},m}(n) = \frac{c_m - 1}{c_m + 1}\frac{ES_m(n-1)}{c_m} + \frac{2}{c_m + 1}\frac{1}{c_m}\frac{ES_m^2(n-1)}{2ES_m(n-1)}$$

   and

$$ET_{M+1}(n) = \big[EL_{M+1}(n-1) + Q_{M+1}(n-1) + p_{M+1}(0|n-1)q\big]\frac{1}{\lambda};$$

   (b)

$$TH(n) = n \Big/ \left(\sum_{r=1}^{R}\sum_{m=1}^{M} \nu_m^{(r)} ET_m^{(r)}(n) + ET_{M+1}(n) \right),$$

$$TH_m^{(r)}(n) = \nu_m^{(r)} TH(n), \quad r = 1, \ldots, R, \ m = 1, \ldots, M;$$

   (c) for $r = 1, \ldots, R$, and $m = 1, \ldots, M$, and for $k = 1, 2$,

$$ES_m^k(n) = \sum_{r=1}^{R} \frac{TH_m^{(r)}(n)}{\sum_{r=1}^{R} TH_m^{(r)}(n)} E\big(S_m^{(r)}\big)^k;$$

   (d) for $m = 1, \ldots, M$, and for $l = 1, \ldots, \min(c_m - 1, n)$,

$$p_m(l|n) = \frac{ES_m(n)}{l}\sum_{r=1}^{R} \nu_m^{(r)} TH^{(r)}(n) p_m(l-1|n-1); \tag{21}$$

   (e) for $m = 1, \ldots, M$, if $n < c_m$, $Q_m(n) = 0$, otherwise,

$$Q_m(n) = \frac{ES_m(n)}{c_m}\sum_{r=1}^{R} \nu_m^{(r)} TH^{(r)}(n)\big[Q_m(n-1) + p_m(c_m - 1|n-1)\big]$$

   and

$$Q_{M+1}(n) = \frac{TH(n)}{\lambda}\big[Q_{M+1}(n-1) + p_{M+1}(0|n-1)q\big];$$

   (f) for $m = 1, \ldots, M$

$$p_m(0|n) = 1 - \sum_{l=1}^{\min(c_m - 1, n)} p_m(l|n) - Q_m(n)$$

and

$$p_{M+1}(0|n) = 1 - Q_{M+1}(n);$$

(g) for $m = 1, \ldots, M$, if $n < c_m$, $E\widetilde{L}_m(n) = 0$, otherwise,

$$E\widetilde{L}_m(n) = \frac{ES_m(n)}{c_m}\nu_m TH(n)\big[E\widetilde{L}_m(n-1) + Q_m(n-1)\big];$$

(h)
$$EL_{M+1}(n) = TH_{M+1}(n)\ ET_{M+1}(n).$$

A.4. Multiclass AMVA

The notation is this appendix is an extended version of that in the previous appendices. In particular, we introduce the vector notation $\vec{n} = (n^{(1)}, \ldots, n^{(R)})$, in which $n^{(r)}$ denotes the number of class r tokens in the network. Furthermore, we denote by $\vec{e}_r$ the R-dimensional vector with a one at position r and a zero at the other positions. Finally, variables depending on the population $\vec{n}$ equal zero if there is a class $r = 1, \ldots, R$ for which $n^{(r)} < 0$.

1. Initialize. Set $p_m(0|\vec{0}) = 1$, $Q_m(\vec{0}) = 0$, $E\widetilde{L}_m(\vec{0}) = 0$, $m = 1, \ldots, M$ and for $r = 1, \ldots, R$, $p_{M+r}(0|\vec{0}) = 1$, $Q_{M+r}(\vec{0}) = 0$, $EL_{M+r}(\vec{0}) = 0$.

2. For $n = 1, \ldots, N$, for all states $\vec{n}$ for which $\sum_{r=1}^{R} n^{(r)} = n$, and $n^{(r)} \leqslant N^{(r)}$ compute:

   (a) for $r = 1, \ldots, R$, and $m = 1, \ldots, M$,

   $$ET_m^{(r)}(\vec{n}) = Q_m(\vec{n} - \vec{e}_r)ES_{\text{rem},m}^{(r)}(\vec{n}) + E\widetilde{L}_m(\vec{n} - \vec{e}_r)\frac{ES_m(\vec{n} - \vec{e}_r)}{c_m} + ES_m^{(r)},$$

   where

   $$ES_{\text{rem},m}^{(r)}(\vec{n}) = \frac{c_m - 1}{c_m + 1}\frac{ES_m(\vec{n} - \vec{e}_r)}{c_m} + \frac{2}{c_m + 1}\frac{1}{c_m}\frac{ES_m^2(\vec{n} - \vec{e}_r)}{2ES_m(\vec{n} - \vec{e}_r)};$$

   and for $r = 1, \ldots, R$,

   $$ET_{M+r}(\vec{n}) = \big[EL_{M+r}\big(\vec{n} - \vec{e}_r\big) + Q_{M+r}\big(\vec{n} - \vec{e}_r\big) + p_{M+r}\big(0|\vec{n} - \vec{e}_r\big)q^{(r)}\big]\frac{1}{\lambda^{(r)}};$$

   (b)
   $$TH^{(r)}(\vec{n}) = n^{(r)}\Big/\left(\sum_{m=1}^{M}\nu_m^{(r)}ET_m^{(r)}(\vec{n}) + ET_{M+r}(\vec{n})\right);$$

   (c) for $r = 1, \ldots, R$, and $m = 1, \ldots, M$, and for $k = 1, 2$,

   $$ES_m^k(\vec{n}) = \sum_{r=1}^{R}\frac{TH^{(r)}(\vec{n})}{\sum_{r=1}^{R}TH^{(r)}(\vec{n})}E\big(S_m^{(r)}\big)^k;$$

(d) for $m = 1, \ldots, M$, and for $l = 1, \ldots, \min(c_m - 1, n)$,

$$p_m(l|\vec{n}) = \frac{ES_m(\vec{n})}{l} \sum_{r=1}^{R} \nu_m^{(r)} TH^{(r)}(\vec{n})\ p_m(l-1|\vec{n} - \vec{e}_r);$$

(e) for $m = 1, \ldots, M$, if $n < c_m$, $Q_m(n) = 0$, otherwise,

$$Q_m(\vec{n}) = \frac{ES_m(\vec{n})}{c_m} \sum_{r=1}^{R} \nu_m^{(r)} TH^{(r)}(\vec{n}) \big[Q_m\big(\vec{n} - \vec{e}_r\big) + p_m\big(c_m - 1\big|\vec{n} - \vec{e}_r\big)\big]$$

and for $r = 1, \ldots, R$,

$$Q_{M+r}(\vec{n}) = \frac{TH^{(r)}(n)}{\lambda^{(r)}} \big[Q_{M+r}\big(\vec{n} - \vec{e}_r\big) + p_{M+r}\big(0\big|\vec{n} - \vec{e}_r\big)\ q^{(r)}\big];$$

(f) for $m = 1, \ldots, M$

$$p_m(0|\vec{n}) = 1 - \sum_{l=1}^{\min(c_m - 1, n)} p_m(l|\vec{n}) - Q_m(\vec{n})$$

and for $r = 1, \ldots, R$,

$$p_{M+r}(0|\vec{n}) = 1 - Q_{M+r}(\vec{n});$$

(g) for $m = 1, \ldots, M$, if $n < c_m$, $E\widetilde{L}_m(n) = 0$, otherwise,

$$E\widetilde{L}_m(\vec{n}) = \frac{ES_m(\vec{n})}{c_m} \nu_m TH(\vec{n}) \big[E\widetilde{L}_m\big(\vec{n} - \vec{e}_r\big) + Q_m\big(\vec{n} - \vec{e}_r\big)\big];$$

(h) for $r = 1, \ldots, R$,

$$EL_{M+r}(\vec{n}) = TH_{M+r}(\vec{n})\ ET_{M+r}(\vec{n}).$$

References

[1] B. Avi-Itzhak and D.P. Heyman, Approximate queuing models for multiprogramming computer systems, Operations Research 21 (1973) 1212–1230.

[2] B. Baynat and Y. Dallery, A product-form approximation method for general closed queueing networks with several classes of customers, Performance Evaluation 24 (1996) 165–188.

[3] A. Brandwajn, Fast approximate solution of multiprogramming models, in: *Performance Evaluation Review* (1982) pp. 141–149.

[4] M.B.N. Bui, Définition d'un outil de modelisation experte, Ph.D. thesis, Université de Pierre et Marie Curie (1989).

[5] R. Buitenhek, Performance evaluation of dual resource manufacturing systems, Ph.D. thesis, University of Twente (1998).

[6] R. Buitenhek, G.J. van Houtum and W.H.M. Zijm, An open queueing model for flexible manufacturing systems with multiple part types and general purpose pallets, Working paper LPOM-96-11, Faculty of Mechanical Engineering, University of Twente (1996).

[7] J.A. Buzacott and J.G. Shanthikumar, Models for understanding flexible manufacturing systems, AIIE Transactions 12 (1980) 339–350.

[8] J.A. Buzacott and J.G. Shanthikumar, *Stochastic Models of Manufacturing Systems* (Prentice Hall, 1993).
[9] K.M. Chandy, U. Herzog and L. Woo, Approximate analysis of queueing networks, IBM Journal of Research and Development 19 (1975) 43–49.
[10] K.M. Chandy, U. Herzog and L. Woo, Parametric analysis of queueing networks, IBM Journal of Research and Development 19 (1975) 36–42.
[11] Y. Dallery, Approximate analysis of general open queueing networks with restricted capacity, Performance Evaluation 11 (1990) 209–222.
[12] E.D. Lazowska and J. Zahorjan, Multiple class memory constrained queueing networks, in: *Performance Evaluation Review* (1982) pp. 130–140.
[13] R. Marie, An approximate method for general queueing networks, IEEE Transactions on Software Engineering 5 (1979) 530–538.
[14] M. Neuts, *Matrix-Geometric Solutions in Stochastic Models – An Algorithmic Approach* (The John Hopkins University Press, 1981).
[15] H.G. Perros, Y. Dallery and G. Pujolle, Analysis of a queueing network model with class dependent window flow control, IEEE InfoCom (1992) pp. 968–977.
[16] M. Reiser and S.S. Lavenberg, Mean-value analysis of closed multichain queueing networks, Journal of the Association for Computing Machinery 27 (1980) 313–322.
[17] L.P. Seelen, An algorithm for PH|PH|c queues, European Journal of Operational Research 23 (1986) 118–127.
[18] M.L. Spearman, D.L. Woodruff and W.J. Hopp, CONWIP: A pull alternative to kanban, International Journal of Production Research 28 (1990) 879–894.
[19] U. Sumita and M. Rieders, Numerical comparison of the replacement process approach with the aggregation-disaggregation algorithm for row-continuous Markov chains, in: *Numerical Solution of Markov Chains*, ed. W.J. Stewart (Marcel Dekker, 1991) pp. 287–302.
[20] A. Thomasian and P. Bay, Analysis of queueing network models with population size constraints and delayed blocked customers, in: *Proceedings of the ACM SIGMETRICS Conference* (Cambridge, 1984) pp. 202–216.
[21] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach* (Wiley, 1986).