MANUFACTURING AND LOGISTIC SYSTEMS ANALYSIS, PLANNING AND CONTROL

Prof. dr W.H.M. Zijm

November 2, 2003
Contents

1 Single stage manufacturing systems .............................. 7
  1.1 Introduction .................................................. 7
  1.2 The Poisson process and PASTA ............................ 7
  1.3 Characterization of queues: Kendall’s notation ............ 8
  1.4 M/M/1 .......................................................... 9
     1.4.1 Deriving \( \pi(n) \) ........................................ 10
     1.4.2 Expressions for the performance measures .......... 11
  1.5 M/G/1 .......................................................... 13
  1.6 M/D/1 queue ................................................... 15
  1.7 G/G/1 .......................................................... 17
  1.8 M/M/c .......................................................... 17
  1.9 G/G/c .......................................................... 20
  1.10 \( M/M/\infty \) and \( M/G/\infty \) ............................ 21
  1.11 Service interruptions ........................................ 22
     1.11.1 Failures ................................................... 22
     1.11.2 Setup Times .............................................. 25
     1.11.3 Rework .................................................... 26
  1.12 Process Batching .............................................. 27
  1.13 Single stage production-to-stock systems .................... 30

2 Job Shop Manufacturing Systems ................................. 33
  2.1 Introduction ................................................... 33
  2.2 Open single class product form networks .................... 33
     2.2.1 A Tandem Queue .......................................... 33
     2.2.2 Jackson-networks ......................................... 36
     2.2.3 Performance measures for Jackson networks .......... 38
     2.2.4 Multiple class Jackson Networks ......................... 39
  2.3 Production to stock models .................................... 40
  2.4 General Job Shop Manufacturing Systems ..................... 41
     2.4.1 Single Class Manufacturing Systems ..................... 41
     2.4.2 Traffic Variability Equations ........................... 42
     2.4.3 Performance measures for general single class manufac-
           turing systems .............................................. 44
     2.4.4 Multi-Class Manufacturing Systems ...................... 45
  2.5 Move batches in general flow lines ........................... 48
### CONTENTS

3 Workload Controlled Manufacturing Systems ........................................... 51
  3.1 Manufacturing systems modelled as Closed Queueing Networks .............. 51
  3.1.1 Gordon-Newell Networks ........................................ 52
  3.1.2 The Convolution Algorithm ...................................... 54
  3.1.3 Mean Value Analysis algorithm for single server station networks .......... 59
  3.1.4 Marginal Distribution Analysis Algorithm .......................... 63
  3.1.5 Multi-Class MDA ............................................... 66
  3.1.6 Restricted Open Queueing Networks ................................ 69
  3.2 General workload controlled manufacturing systems ........................... 70
    3.2.1 Single Class WCMS ......................................... 70
    3.2.2 Multi-class WCMS ........................................... 72

4 PAC systems ............................................................................. 77
  4.1 Introduction ........................................................................ 77
    4.1.1 Production to order PAC systems ................................ 77
    4.1.2 Overview of this chapter ........................................ 79
  4.2 Exact analysis .................................................................... 79
    4.2.1 The Markov process ................................................. 79
    4.2.2 The stationary distribution ....................................... 81
    4.2.3 The stability condition ............................................ 83
  4.3 Approximating production to order models ...................................... 84
    4.3.1 Evaluation of the CQN without the synchronization station .......... 85
    4.3.2 Constructing a load-dependent server as a substitute for
        the synchronization station ........................................ 86
    4.3.3 Evaluation of the CQN-LD including the synchronization
        station ................................................................. 88
    4.3.4 The Algorithm ...................................................... 89
  4.4 Production to stock models .................................................... 89
  4.5 Multi-Class Networks: universal cards ........................................ 91
    4.5.1 The model ......................................................... 92
    4.5.2 Complete reduction .............................................. 93
  4.6 Multi-Class Networks: dedicated cards .......................................... 94
    4.6.1 The model ......................................................... 94
    4.6.2 The extended PDP method ....................................... 96
Introduction

In this course, we will develop a number of analytical models for manufacturing and logistic systems that may help to improve the design, planning and control of such systems. Typical design questions concern e.g. the effective capacity of a manufacturing system, i.e. what is the maximum throughput that a manufacturing system can handle, given a certain product mix? Other highly important questions relate to customer order lead time: How long does a customer have to wait before a promised order can be delivered? What lead time variability can be expected, given a particular workload and order mix? Should the production line produce to order (typical for highly customized products) or to stock (typical for mass production)? These questions do not just concern the manufacturing system but relate to the design of the logistic environment as a whole. Still other questions are addressed regarding the quality of products and the flexibility of the production system: how easy can we change from production of one product range to another one? A thorough understanding of reliability issues and maintenance policies further helps to design efficient and effective manufacturing and logistic systems.

Design questions arise in situations in which the day-to-day order mix is generally unknown, or even the product range to be produced is not completely specified. Besides, due to technological innovations, some products may phase out and others be developed for production. Hence, in general, product routings through a factory and job processing times are seldom precisely known in a design stage. Also, demand for products may vary over time, both with respect to volume and mix. For these reasons, modeling a production system as a network of stochastic processes during the design phase seems to be a logical choice.

In this course, we model a workstation, consisting of a number of basically identical machines, as a single-stage, multi-server queueing station. A part or subassembly that such a workstation processes, is called a job. Usually, jobs require processing at multiple workstations performing operations such as turning, milling, drilling, forming, assembly and testing. In such a case, the various workstations may be seen as linked through the job’s process routings. Clearly, different jobs may have different routings. Therefore, we model the manufacturing system as a queueing network, since these models allow for uncertainty both with respect to job routings and processing times. In the same way, a logistic distribution system may be seen as a chain, consisting of several sub-networks (manufacturing systems and distribution warehouses), subject to uncertain demand.

After the design and implementation of a production line, shorter term questions arise relating to planning and control, such as: How many products do we
have to produce next month? How to schedule a set of known customer orders in a production system such that promised customer order delivery dates are met? What is the optimal batch size, given a specific list of customer orders? Or, in a production-to-stock environment: when to re-order a batch of specific items in order to keep an inventory in a warehouse high enough to fulfill expected demand in the next four weeks? Such questions are typically addressed by Manufacturing Resources Planning systems. These systems mostly focus on lead time planning in discrete manufacturing systems, more specifically in assembly situations, but have proven to be inadequate in situations where tight capacity restrictions are important. Therefore, we offer an alternative in this course, in that we simultaneously address lead time and capacity management issues, in one overall framework. For the short term, with more certainty on product routings and processing times, we typically rely on mathematical programming models, in particular on linear and integer programming models and on combinatorial optimization methods, such as network flow, matching and scheduling algorithms.

This course is divided into two parts. In the first part, we develop a queueing framework to model fairly general manufacturing and logistic systems. We will treat both production-to-order and production-to-stock models, as well as mixtures of these two basic systems (e.g. component production to stock, (customized) final product assembly to order). We will extend the models of manufacturing systems to complete logistic networks, covering both supply of raw materials, production, and distribution of products to a variety of locations. In the second part, we will discuss deterministic optimization models to cover both mid term capacity planning issues as well as the scheduling and control of short term production. Again, we distinguish between production-to-order, dealing with issues such as due date satisfaction, and production-to-stock, where we use capacitated inventory models to guarantee a sufficiently high customer order service level.
Chapter 1

Single stage manufacturing systems

1.1 Introduction

In a general manufacturing system jobs do not arrive in a regular fashion, neither do they all require the same amount of service. Therefore job interarrival and service times are described by random variables that have a specified distribution. Generally it is assumed that the arrivals and service requests are independent and identically distributed, i.i.d., random variables. Another important constraint is that most queueing systems we will consider are work conserving. This means that a server, as long as there are jobs in the system, will be busy processing jobs.

1.2 The Poisson process and PASTA

Given that the interarrival times have some arbitrary distribution, we would like to know the number of arrivals, \( N(T) \), we can expect to have in a certain interval \( T \). More generally, we want to relate the interarrival distribution to the distribution of \( N(T) \). This appears to be possible for exponentially distributed interarrival times with parameter \( \lambda \).

Define \( T_n \) as:

\[
\sum_{i=1}^{n} X_i,
\]

with \( X_i \) is the interarrival time between job \( i-1 \) and job \( i \). Another quantity of interest is

\[
N(t) = \max(n : T_n \leq t),
\]

that is, the number of arrivals that will have occurred before or at \( t \). Obviously we have

\[
\mathbb{P}(N(t) \geq n) = \mathbb{P}(T_n \leq t).
\]

The distribution of the sum of \( n \) i.i.d. exponential random variables is Erlang
distributed with parameters $n$, and $\lambda$, see [66], which means that

$$P(T_n \leq t) = \int_0^t \lambda e^{-\lambda t} \frac{(\lambda t)^{n-1}}{(n-1)!} dt.$$ 

Using partial integration and the fact that for any $x \in \mathbb{R}, x^n/n! \to 0$, if $n \to \infty$, we find

$$P(T_n \leq t) = \sum_{j=n}^{\infty} e^{-\lambda t} \frac{e^{\lambda t} \lambda^j}{j!}.$$ 

From this and eq: 1.1, it follows that

$$P(N(t) = n) = P(N(t) \geq n) - P(N(t) \geq n + 1) = e^{-\lambda t} \frac{(\lambda t)^n}{n!}.$$ 

The distribution we now have obtained for $N(t)$ is known as the Poisson distribution. For this reason the job arrival process when the interarrival times are exponential is often referred to as a Poisson process.

**Ex. 1.2.1** Show that $E(N(t)) = \lambda t$ and $\text{Var}(N(t)) = \lambda t$.

Poisson arrivals satisfy a remarkable property called ‘Poisson Arrivals See Time Averages’, or briefly, ‘PASTA’, see [86]). This means that when jobs arrive according to a Poisson process they will ‘see’ the system as if it is in equilibrium, assuming that such a state exists. The probability measure that describes the system’s state is related to time, that is, the fraction of Poisson arrivals that observe the system in state $E$, say, is equal to the fraction of time that the system spends in state $E$.

1.3 Characterization of queues: Kendall’s notation

A queueing process depends on the interarrival and service time distribution of jobs, the number of servers, and the available buffer space. These factors are so fundamental that a specific notation to characterize queues has developed that only depends on these factors. This is called Kendall’s notation. (See [22] for a more detailed description.)

We speak of an $A/B/n/K$ queue, when the interarrival, resp. service distribution is of type A, resp. B. The number of servers is indicated by $n$, the number of buffer locations by $K$. In general, when $K$ does not appear in the queue’s specification, it is implicitly assumed that the buffer size is infinite. In this report we will not be concerned with phenomena related to buffer overflow. Thus we will never explicitly mention $K$.

The two most important interarrival and service time distributions are the exponential and the general distribution. The former is indicated by an $M$ to reflect the ‘Markovian’, or ‘Memoryless’, property of the exponential distribution. The latter is simply $G$. Of course, the $G$ can be further specified. For instance, the letter $D$ is used to denote a deterministic arrival process.

The most well known queue is the $M/M/1$ queue. It has one server whose jobs arrive according to a Poisson process and require an exponentially distributed amount of service. Furthermore its buffer space is unlimited. Another
1.4. M/M/1

One of the easiest queueing systems to analyze is the M/M/1 queue. In this system one machine services jobs that have exponentially distributed interarrival times and service times. It is common to denote the parameter of the interarrival time distribution by $\lambda$ and the parameter of the service time by $\mu$.

The behavior of the M/M/1 queue can be described by a Markov process with continuous time and discrete states $n \in \mathbb{N} \cup \{0\}$, where $n$ is the number of jobs present in the system (= the number of jobs in queue and being serviced, if any). Figure 1.1 shows a sample path of the amount of work in the system.

Let $p(n,t)$ denote the probability that the server contains $n$ jobs at time $t$. When the Markov process is stable, the limit $\lim_{t \to \infty} p(n,t)$ exists, suggesting the definition

$$\pi(n) = \lim_{t \to \infty} p(n,t),$$

(1.4)

to denote the probability in stationary state. This means that the fraction of time the server contains $n$ jobs is equal to $\pi(n)$. Clearly $\pi(n)$ should, and in fact does, not depend on starting conditions.

Many performance measures of interest can be defined in terms of $\pi(n)$. The expected number of jobs in the system, or ‘work in process’, is:

$$\mathbb{E}L = \sum_{n=0}^{\infty} n \pi(n).$$

(1.5)

The expected time in the system, also called ‘cycle time’ or ‘flow time’, can be found with Little’s formula:

$$\mathbb{E}L = \lambda \mathbb{E}W.$$

(1.6)
The waiting time in the queue $E_W$ can now be found from the equality:

$$E_W = E_{W_Q} + E_S,$$

(1.7)

where $E_S = 1/\mu$ is the expected service time of a job. In words this means that a job spends $E_{W_Q}$ in queue before its service starts, and then receives $E_S$ units of service. Altogether the total time in the system should be the sum of these two terms.

Another important quantity in queueing theory is the system’s utilization $\rho$:

$$\rho = \frac{\lambda}{\mu} = \lambda E_S,$$

(1.8)

For a system to be stable it is necessary that $\rho < 1$, otherwise jobs arrive, on the average, faster to the system, than they can be processed. In other words: the system will only be stable when the rate at which work can be processed is higher than the rate at which it arrives, i.e. $(\mu > \lambda)$. On the other hand, it is intuitively clear that when $\rho \geq 1$ the number of jobs in the system will increase without bound with probability 1.

### 1.4.1 Deriving $\pi(n)$

We will now derive the stationary state probabilities $\pi(n)$ by considering global balance equations. These state that, for stable\(^1\) systems, the rate at which a process enters a certain state should equal the rate at which it departs from that state. This means for the M/M/1 queue that

$$\text{rate out} = \text{rate in}$$

$$\lambda \pi(0) = \mu \pi (1)$$

(1.9)

$$(\lambda + \mu) \pi(n) = \lambda \pi(n - 1) + \mu \pi(n + 1),$$

(1.10)

for $n = 1, 2, \ldots$. Note that here all transitions into and out of a state are considered. Figure 1.2 shows these states and transition rates graphically.

**Ex. 1.4.1** Show that the global balance equations (1.9, 1.10) reduce to

$$\lambda \pi(n) = \mu \pi(n + 1) \quad n \geq 0.$$

(1.11)

These equations are often called local balance equations as they indicate that the ‘flow’ out of $\pi(n)$ to $\pi(n + 1)$ balances the flow from $\pi(n + 1)$ to $\pi(n)$. Note that the local balance equations do not always enable one to solve the Markov chain.

\(^1\)To be precise, the Markov chain should be ‘ergodic’, which means that: every state in the chain will be accessed infinitely often; the time between two visits is finite with probability one; and finally that it is aperiodic. See [38, 67] for further detail.
The equations (1.11) allow us to obtain $\pi(n)$ straightaway,

$$\pi(n + 1) = \frac{\lambda}{\mu}\pi(n) = \rho \pi(n), \quad n \geq 0.$$  
(1.12)

It follows that we can express $\pi(n)$ in terms of $\pi(0)$,

$$\pi(n) = \pi(0) \rho^n.$$  
(1.13)

Using the ‘law of total probability’,

$$\sum_{n=0}^{\infty} \pi(n) = 1,$$  
(1.14)

we obtain for $\pi(0)$, provided the system is stable, i.e., $\rho < 1$,

$$\pi(0) = 1 - \rho,$$

so that,

$$\pi(n) = (1 - \rho)\rho^n,$$  
(1.15)

Note that the utilization, defined as the probability that the station is busy, i.e., contains at least one job, is simply

$$\sum_{k=1}^{\infty} \pi(n) = 1 - \pi(0) = \rho.$$  
(1.16)

Often $\pi(n)$ is written with a normalization constant $G$, as follows,

$$\pi(n) = \frac{1}{G} \rho^n,$$  
(1.17)

so that

$$G = \frac{1}{\pi(0)} = \frac{1}{1 - \rho}.$$  
(1.18)

### 1.4.2 Expressions for the performance measures

Now we can find an expression for the expected number of jobs in the system, and the other measures as defined above.
We start with
\[ E_L = \sum_{n=0}^{\infty} n \pi(n) \]
\[ = (1 - \rho) \sum_{n=0}^{\infty} n \rho^n \]
\[ = (1 - \rho) \rho \sum_{n=1}^{\infty} n \rho^{n-1} \]
\[ = (1 - \rho) \rho \frac{d}{d\rho} \left[ \sum_{n=0}^{\infty} \rho^n \right] \]
\[ = (1 - \rho) \rho \frac{d}{d\rho} \left[ \frac{1}{1 - \rho} \right] \]
\[ = (1 - \rho) \rho \frac{1}{(1 - \rho)^2} \]
\[ = \frac{\rho}{1 - \rho}, \quad \rho < 1. \] (1.19)

Note that the expected number of jobs in the system only depends on the utilization.

To calculate the expected time in the system \((E_W)\) we make use of Little’s law eq. (1.6) to get:
\[ E_W = \frac{\rho}{1 - \rho} \frac{1}{\lambda} \]
\[ = \frac{1}{1 - \rho} \frac{1}{\mu}, \quad \rho < 1. \] (1.20)

From formula (1.20) it is easy to see that when the utilization tends to 1 the expected cycle time will grow without bound.

For the M/M/1 queue the distribution of the time in the system can be derived as well. This turns out to be exponentially distributed with parameter \(\mu - \lambda\), see [66] for a proof.

The expected waiting time in the queue can be found according to:
\[ E_{WQ} = E_W - E_S \]
\[ = \frac{1}{1 - \rho} \frac{1}{\mu} - \frac{1}{\mu} \]
\[ = \frac{\rho}{1 - \rho} \frac{1}{\mu}, \quad \rho < 1. \] (1.21)

This can also be found by making use of the PASTA property, which is valid as here the arrival process is Poisson. An arriving job will observe the system as if it is in stationary state. Hence the job will, at its arrival, expect to see \(E_L\) jobs before it in the queue. Each of these jobs will require on average \(E_S\) units of work.

When we apply Little’s law to the expected waiting time we can find an expression for the expected number of jobs in the queue:
\[ E_{LQ} = \lambda E_{WQ} = \frac{\rho^2}{1 - \rho}. \] (1.22)
Note that this equation can also be obtained by subtracting $\rho$ jobs from the expected number of jobs in the system. This can be seen by noting that $\rho = 1 - \pi_0$ is the probability that there is at least one customer in the system, and for work conserving queues such as the M/M/1 queue, this is equal to the expected number of jobs to be in service (Think about this!).

**Example 1.1**

We have a repair shop in which broken items are repaired. The manager is asked whether the number of jobs that the shop repairs per day can be increased.

To answer this question the manager assumes that the items arrive according to a Poisson process, and that the service times are exponentially distributed. Now she should find values for the arrival rate $\lambda$ and the service rate $\mu$. She estimates the number of arriving faulty items at 5 per day. Estimating the service time is more complicated as the items require varying sorts of repair, all of which have different repair times. Thus she decides to proceed in another way, that is, to count the number of items in the shop at each arrival of a faulty item.

**Ex. 1.4.2** Is this a good strategy for acquiring statistically sound data about the expected number of items, $E(L)$ in the shop? What are the implications of PASTA?

**Ex. 1.4.3** How would you estimate $E(L)$ with these data?

**Ex. 1.4.4** Can you think of a method to determine how many arrivals she needs such that

$$\frac{1}{N} \sum_{i=1}^{N} L_i,$$

where $L_i$ is the number of items at the $i$-th arrival, is a good estimator for $E(L)$?

She obtains $E(L) = 10$. From this she computes $\rho = 10/11$. Suppose the new arrival rate would be 7 per day. This would be too much, for:

$$\rho' = \frac{\lambda'}{\lambda} \rho = \frac{7}{5} \frac{10}{11} > 1.$$

Interestingly, only the relative increase of workload is important to determine system stability. The actual arrival rate plays no role here.

### 1.5 M/G/1

In practice it is often the case that the service time distribution is not well described by the exponential distribution. Hence we need to derive performance measures for more general queueing systems in which service times can have an arbitrary distribution. Such queues are denoted as M/G/1 queues, where G stands for a general distribution. As may be expected, the M/G/1 queue is harder to analyze than the M/M/1 queue because the service time distribution is no longer memoryless. For instance, there are no simple expressions for the
probabilities $\pi(n)$. Still we can find the expected waiting time in the system and the queue. The expression for $\mathbb{E}W_Q$ is generally know as the Pollaczek-Khintchine, or PK, formula. Deriving this will be the objective of this section.

The proof is quite simple if we use a basic and intuitive ‘cost’ identity, [66]. This cost principle is interesting in its own right, and can be applied to derive other queueing identities such as Little’s formula. Imagine that jobs entering a queueing system, whether it is a single server queue or a complete network, have to pay money to the system according to some general rule. Then we find that, see [66] for a heuristic proof,

$$\mathbb{E}(\text{earning rate of system}) = \lambda \mathbb{E}(\text{amount of money each job pays}).$$

This cost identity is the queueing analog of the principle of ‘conservation of money’, or stated differently, ‘What goes in, goes out, on the average’.

To derive the PK formula we make a smart choice for the payment rule, as follows. ‘Each job pays at a rate of $y$ per unit time, when its remaining service time is $y$, whether he is in queue or in service’. More specifically, when a job is being served and still requires $y$ units of service, its contribution to the amount of work in the system is $y$. Therefore he pays at a rate of $y$ per unit time. A job in queue demanding $S$ units of service, should pay $S$ per unit time (as long as it is not in service). Now the system will earn at a rate that is equal to the amount of work, $V$, that is in the system. By the cost identity we therefore get:

$$\mathbb{E}V = \lambda \mathbb{E}(\text{amount of money each job pays}).$$

A job entering the system with $S$ units of work will pay at a rate of $S$ while in queue, and $S - x$ after having received $x$ units of service. Then we have:

$$\mathbb{E}(\text{amount of money each job pays}) = \mathbb{E} \left[ (SW_Q) + \int_0^S (S - x) \, dx \right],$$

and thus,

$$\mathbb{E}V = \lambda \mathbb{E}(SW_Q) + \lambda \frac{ES^2}{2}. \quad (1.23)$$

If the service time $S$ is independent of the waiting time in queue (as is generally the case), we arrive at:

$$\mathbb{E}V = \lambda \mathbb{E}W_Q \mathbb{E}S + \lambda \frac{ES^2}{2}.$$ 

Due to the PASTA property, an arriving job will see a single server system in equilibrium, and will expect to see $\mathbb{E}V$ units of work in front of it, before its service can start (assuming FIFO scheduling). Hence, for $M/G/1$ queues,

$$\mathbb{E}(\text{the job’s waiting time in the queue}) = \mathbb{E}(\text{work in the system as seen by an arrival}),$$

which means that

$$\mathbb{E}W_Q = \mathbb{E}V.$$

This result together with (1.23) gives the Pollaczek-Khintchine formula,

$$\mathbb{E}W_Q = \frac{\lambda}{2} \frac{ES^2}{1 - \rho}. \quad (1.24)$$
where we defined

\[ \rho = \lambda \mathbb{E}S. \]

**Ex. 1.5.1** What result do you obtain when you apply the cost identity to the rule according to which a job pays $1 per unit time while he is served, and nothing otherwise, i.e., in queue?

We can write the PK formula as well in terms of the squared coefficient of variation (SCV) of the service time, \( C_s^2 \),

\[ C_s^2 = \frac{\sigma_s^2}{(\mathbb{E}S)^2}, \tag{1.25} \]

with \( \sigma_s^2 \) the variance of the service time, so that

\[ \mathbb{E}W_Q = \frac{1}{2}(1 + C_s^2) \frac{\rho}{1 - \rho} \mathbb{E}S, \quad \rho < 1. \tag{1.26} \]

(Check that the two are equivalent).

The expected time in the system can be found with:

\[ \mathbb{E}W = \mathbb{E}W_Q + \mathbb{E}S \\
= \frac{1}{2}(1 + C_s^2) \frac{\rho}{1 - \rho} \mathbb{E}S + \mathbb{E}S, \quad \rho < 1, \tag{1.27} \]

and by applying Little’s law we can find an expression for the expected number of jobs in the system:

\[ \mathbb{E}L = \frac{1}{2}(1 + C_s^2) \frac{\rho^2}{1 - \rho} + \rho, \quad \rho < 1. \tag{1.28} \]

Note that for an M/M/1 queue \( C_s^2 = 1 \). The reader is asked to verify that in this case equations (1.26), (1.27) and (1.28) reduce to (1.21), (1.20) and (1.19), respectively.

### 1.6 M/D/1 Queue

An interesting example of the M/G/1 queue is the M/D/1 queue where the service times are deterministic. Since the service times are deterministic the variance of the service times \( \sigma_s^2 = 0 \) and therefore also the squared coefficient of variation \( C_s^2 = 0 \). When we substitute this into equation (1.26) we find:

\[ \mathbb{E}W_Q(M/D/1) = \frac{1}{2} \frac{\rho}{1 - \rho} \mathbb{E}S \\
= \frac{1}{2} \mathbb{E}W_Q(M/M/1), \quad \rho < 1. \tag{1.29} \]

as \( \frac{\rho}{1 - \rho} \mathbb{E}S \) is the expected waiting time of the M/M/1 queue. Applying Little’s result, we see that the M/D/1 queue is half the length of the M/M/1 queue for systems with equal utilization and mean service times.
Example 1.2

The manager of our previous example is asked to reduce the number of items in repair in the shop, that is, to increase the ‘throughput’. In the next few examples we will work out and compare various strategies to realize this.

Her first idea is to simply throw away items that have not been repaired within some fixed amount of time $T$, and replace these items with new ones. The service distribution will now become somewhat more complicated however.

Ex. 1.6.1 Can you sketch the probability density function of the new service distribution, given that the initial distribution is exponential. Hint: the items that are thrown away require exactly $T$ units of service.

The manager decides to replace the ‘chopped off’ service distribution by an upper bound:

$$P(S \leq x) = \begin{cases} 0 & x < T, \\ 1 & x \geq T. \end{cases}$$

Ex. 1.6.2 Why is this an upper bound for the service distribution?

Ex. 1.6.3 Why is the squared coefficient of service variation for this distribution equal to 0?

Ex. 1.6.4 Compare $E_L$ of this queue to the same expression for the M/M/1 queue.

Ex. 1.6.5 The manager does not want to reduce the workload. What should $T$ be such that the utilization in the M/D/1 queue equals 10/11? What fraction of items will be replaced?

$E_L$ can be easily found in this situation, and equals roughly 5.45, assuming $\rho = 10/11$. The manager concludes that indeed a reduction of the variation in the service time makes the number of items in the shop smaller.

Suppose the initial service distribution is uniform over the interval $(0, A]$, i.e.

$$P(S \leq x) = \frac{x}{A}, \quad 0 < x \leq A,$$

instead of exponential.

Ex. 1.6.6 What are $E_{W_Q}$, $E_W$ and $E_L$ in this case? What do you conclude if you compare these results to those as derived for the M/M/1 case?

Ex. 1.6.7 If the manager accepts to dispose of 1 out of 4 items, what should the ‘cut off’ time $T$ be? What is the effect on $E_{W_Q}$, $E_W$ and $E_L$? Is it as interesting to throw away the more difficult items as it was in the M/M/1 case? Why (not)?
1.7 G/G/1

In this section we also dispose of the assumption that the interarrival times are exponentially distributed, because in most real-world manufacturing systems this is not true. The resulting queueing process, the G/G/1 queue, is much harder to analyze than the M/G/1 queue.\(^2\) Only approximations are known for most performance measures, the utilization being an exception to this when we define it as

\[ \rho = \frac{E_S}{EA}, \]

where EA is the expected interarrival time of jobs. We will define for this queue the arrival rate \( \lambda = 1/EA \), analogous to the case when interarrivals have exponential distribution.

Still we can obtain a, for practical purposes, sufficiently accurate approximation for \( E_WQ \). This estimate is based on the first and second moments of the interarrival and service times, more precisely, the squared coefficients of variations of the service and interarrival times. In the previous section we have already introduced the squared coefficient of variation of the service times. Likewise we have for the squared coefficient of variation of the interarrival times:

\[ C^2_a = \frac{\sigma^2_a}{(EA)^2} = \sigma^2_a \lambda^2, \quad (1.30) \]

where \( \sigma^2_a \) is the variance of the interarrival time.

The following expression for the expected waiting time in the queue was proposed by Kingman [43]:

\[ E_WQ = \frac{C^2_a + C^2_s}{2} \frac{\rho}{1 - \rho} E_S, \quad \rho < 1. \quad (1.31) \]

This approximation is reasonably accurate for typical manufacturing systems, except when \( C^2_s \) and \( C^2_a \) are much larger than one or when \( \rho < 0.1 \) (light traffic conditions), or \( \rho > 0.95 \) (heavy traffic).

In the same way as in the previous section we find for the expected time in the system:

\[ E_W = \frac{C^2_a + C^2_s}{2} \frac{\rho}{1 - \rho} E_S + E_S, \quad \rho < 1, \quad (1.32) \]

and for the expected number of jobs in the system:

\[ E_L = \frac{C^2_a + C^2_s}{2} \frac{\rho^2}{1 - \rho} + \rho, \quad \rho < 1. \quad (1.33) \]

The reader should check that these results reduce to those found for the M/G/1 queue where the interarrivals times are exponential.

1.8 M/M/c

In the previous sections we have only discussed single server queueing systems. In this section we handle multiple identical machines at a station with exponentially distributed interarrival and service times. This queue is denoted as an

\(^2\)The key observation is that no longer an embedded Markov chain can be found, as is the case for the M/G/1 queue. See Kleinrock [45] for further detail.
M/M/c queue, with \( c \) the number of parallel servers. The M/M/c queue can be described by a Markov process; Figure 1.3 gives the state transition diagram of this queueing system. It is important to observe that the state transitions of \( k \) jobs to \( k - 1 \) jobs, denoted as \( q_{k,k-1} \), are not the same as for single server queues, but have the form

\[
q_{k,k-1} = \begin{cases} 
    k\mu & k < c, \\
    c\mu & k \geq c.
\end{cases}
\]

If we define the utilization on the station to be

\[
\rho = \frac{\lambda}{c\mu}, \quad \rho < 1, \quad (1.35)
\]

we arrive at the following result for the steady state probabilities of the number of jobs in the system (which the reader should try to derive):

\[
\pi(n) = \frac{1}{G} \frac{(cp)^n}{n!}, \quad n = 1, 2, \ldots, c - 1, \quad (1.36)
\]

and

\[
\pi(n) = \frac{1}{G} \frac{c\rho^n}{c!}, \quad n = c, c + 1, \ldots \quad (1.37)
\]

The normalization constant \( G \) can again be found by filling in \( \sum_n \pi(n) = 1 \).

\[
G = \sum_{n=0}^{c-1} \frac{(cp)^n}{n!} + \sum_{n=c}^{\infty} \frac{c\rho^n}{c!}
\]

\[
= \sum_{n=0}^{c-1} \frac{(cp)^n}{n!} + \frac{(cp)^c}{c!} \sum_{n=0}^{\infty} \rho^n
\]

\[
= \sum_{n=0}^{c-1} \frac{(cp)^n}{n!} + \frac{(cp)^c}{(1 - \rho)c!}. \quad (1.38)
\]

Again we want to determine the expected number of jobs in the system, in queue, and the expected time in the system. For the average number of jobs in
the queue we can find:

\[ E_{\text{LQ}} = \sum_{n=c+1}^{\infty} (n - c) \pi(n) \]
\[ = \sum_{n=c+1}^{\infty} (n - c) \frac{c^c \rho^n}{c!} \frac{1}{G} \]
\[ = \frac{c^c}{c!G} \sum_{n=c+1}^{\infty} (n - c) \rho^n \]
\[ = \frac{c^c}{c!G} \sum_{n=0}^{\infty} (n + 1) \rho^{n+c+1} \]
\[ = \frac{(cp)^c}{c!G} \rho \sum_{n=0}^{\infty} (n + 1) \rho^n \]
\[ = \frac{(cp)^c}{c!G} \frac{\rho}{(1 - \rho)^2}, \quad \rho < 1. \]  

Clearly this expression only depends on the workload \( \rho \) and the number of servers \( c \). For the expected number of jobs in service we have:

\[ E_{\text{LS}} = \sum_{n=0}^{c-1} n \pi(n) + \sum_{n=c}^{\infty} c \pi(n) = cp = \frac{\lambda}{\mu}. \]  

Note that this also could be obtained directly from the definition of the utilization. The expected number of jobs at the station is now easy:

\[ E_{\text{L}} = E_{\text{LQ}} + E_{\text{LS}} \]
\[ = \frac{(cp)^c}{c!G} \frac{\rho}{(1 - \rho)^2} + cp. \]  

Applying Little provides the expected time in the system:

\[ E_T = \frac{E_{\text{L}}}{\lambda} = \frac{(cp)^c}{c!G} \frac{1}{(1 - \rho)^2} \frac{1}{c \mu} + \frac{1}{\mu}. \]  

and by subtracting the expected service time we find for the expected waiting time in the queue:

\[ E_{WQ} = \frac{(cp)^c}{c!G} \frac{1}{(1 - \rho)^2} \frac{1}{c \mu}. \]  

These equations are all quite elaborate and do not provide much intuitive insight for the system. An easier to use approximation of the expected waiting time is proposed in [68]:

\[ E_{WQ} = \frac{\rho (\sqrt{2c(c+1)} - 1)}{c (1 - \rho)} E_{\text{S}}. \]  

Example 1.3
Figure 1.4: Expected number of items in repair as a function of the utilization, computed in the case of an M/M/5 queue

Our manager realizes that throwing away items can be quite costly. She might as well employ an extra repair man to reduce the queue length. If the queue reduction is sufficient, this alternative could be cheaper. Thus she is faced with computing the impact of an extra repair man.

The manager decides to refine the previously used models, and to use a multi server queue model, with initially 5 servers (one for each repair man), and then recompute $E_L$ for 6 servers. To this aim she solves for $\rho$ in $E_L(\rho) = 10$, where the dependency of $E_L$ on $\rho$ is made explicit. This is most easily done by means of the graph of $E_L$, see figure 1.4. She reads out that with five men, $\rho = 0.88$ per employee.

**Ex. 1.8.1** What arrival process will each employee see if the arrival stream at the shop, which is assumed to be Poisson, is split evenly over all men?

**Ex. 1.8.2** Suppose each server in an M/M/c queue, with $c > 1$, works at rate $\mu$. If you compare this to an M/M/1 queue in which the server works with rate $c\mu$, which of the two would build up the larger queue, given that jobs arrive at rate $\lambda$ at each system?

**Ex. 1.8.3** Why is the workload per employee slightly less than $10/11$, where $10/11$ was the workload of the shop modeled as an M/M/1 queue?

**Ex. 1.8.4** Given a workload per employee of 0.88 when there are 5 men, check that $E_L = 5.4$ when 6 men work in the shop.

### 1.9 G/G/c

As mentioned before in real-world manufacturing systems the interarrival and service times seldom have an exponential distribution. The G/G/c queue de-
scribes more general situations. Again, as with the $G/G/1$ queue no exact solution can be found so that we have to be satisfied with approximations.

When we look at equation (1.31) for the $G/G/1$ queue we notice that it can also be presented as follows:

$$E_W(Q(G/G/1) = C^2 + C^2 \rho \cdot \rho < 1,$$ (1.45)

as $\frac{C^2}{\rho}$ equals the expected waiting time of a job in the $M/M/1$ queue. This suggests the following approximation for the $G/G/c$ queue:

$$E_W(Q(G/G/c) = C^2 + C^2 \rho \cdot \rho < 1,$$ (1.46)

In the previous section we have found two equations for the expected waiting time of the $M/M/c$ queue, equation (1.43) is exact and equation (1.44) is an approximation. Both equations can be used in equation (1.46) but we need to keep in mind that equation (1.44) is less accurate but also easier to use and computationally less demanding. From equation (1.46) we again can find equations for the expected time in the system and the expected number of jobs in the system.

1.10 $M/M/\infty$ and $M/G/\infty$

The limit $c \to \infty$ of the $M/M/c$ queue yields the $M/M/\infty$ queue. This type of server station is known as an ample server. Ample servers do not necessarily have to have an infinite number of servers. The crucial point is that there should be enough capacity such that jobs always can be serviced immediately, i.e., without having to wait in a queue. An example of such a server is a continuous transport system such as a conveyor belt.

Having motivated our interest for this type of queue, we derive the state probabilities. Note first that the limit $c \to \infty$ of $G$ in eq. (1.38) is straightforward, and yields:

$$G = \sum_{n=0}^{\infty} \frac{1}{n!} \left( \frac{\lambda}{\mu} \right)^n = e^{\lambda/\mu}.$$

Hence we have, by eq. (1.36), that

$$\pi(n) = e^{-\frac{\lambda}{\mu}} \frac{(\frac{\lambda}{\mu})^n}{n!}.$$ (1.47)

Ex. 1.10.1 The second term in eq. (1.38) becomes 0 when $c \to \infty$. Why?

Ex. 1.10.2 Show that this queue is stable, no matter the value of $\lambda$ (as long as it is finite.)

Ex. 1.10.3 Compute $E_L$ for the $M/M/\infty$ queue.

It is a remarkable fact that the same expressions for $\pi(n)$ hold when the service is of general type. This queue is called the $M/G/\infty$ queue. The reader may consult [67] for a proof.
1.11 Service interruptions

In this section we deal with machine failures, setup times and rework. These phenomena impact the mean and variability of machine service times as they can be seen as outages of machines in the system. Clearly, while a machine is under repair or is being setup, it cannot process jobs. Rework can also be interpreted as an outage when we realize that during rework the machine cannot service new jobs.

We have to distinguish between preemptive and non-preemptive outages. Preemptive outages can occur while a machine is busy processing a job (it ‘pre-empts’ the server.). Power outages, or break down of a cutting tool are examples of this type of machine service suspension. Setups and rework are typical non-preemptive outages, e.g. when the mold of an injection molding machine has to be changed the machine needs to be empty.

It is possible that both preemptive and non-preemptive outages occur simultaneously in manufacturing systems. These mixed scenarios require a more careful analysis than we present here. [39] provide further insight.

1.11.1 Failures

In manufacturing systems all machines eventually fail. These failures increase the waiting time in queues because while the machine is being repaired no jobs can be serviced. We will now study how the working and repair phases of a single machine influences the expected waiting time and the variability, assuming that failures, repairs, job arrivals and services are all independent random events. We furthermore assume that a machine in good shape may fail after an exponentially distributed time. Finally we will restrict our analysis to the situation in which jobs, that are hit by a failure, will not lose the amount of service they received before the failure. This service type is known as preemptive resume.

First we introduce some vocabulary. When a machine is under repair, it is ‘down’ during an interval $D$. Let $m_r = \mathbb{E}D$, hence $m_r$ is the mean time to repair (MTTR). Between two consecutive down times, the machine is ‘up’– it works–for an interval $U$. Let $m_f = \mathbb{E}U$, hence $m_f$ is the mean time to fail (MTTF). The availability, $A$, of the machine is now defined as

$$A = \frac{m_f}{m_f + m_r}. \quad (1.48)$$

To take the impact of failures on a job’s time in the system into account, we define the effective service $S_e$ of a machine as:

$$S_e = S_0 + \sum_{i=1}^{N} D_i,$$

where $D_i$ denotes a $i$-th down time, $N$ is the number of failures that occurred during the service of a job, and $S_0$ is the service duration without failures. Figure 1.5 gives a graphical representation.

$N$ depends on $S_0$ (why?), whereas $D_i$ and $S_0$ are independent random variables by assumption. As stated, we want to find $\mathbb{E}S_e$. Conditioning on $S_0$ shows:

$$\mathbb{E} \left( \sum_{i=1}^{N} D_i \mid S_0 \right) = \mathbb{E}N \mid S_0 \mathbb{E}D.$$
1.11. SERVICE INTERRUPTIONS

Figure 1.5: The upper horizontal line is the time the jobs spend at the server, whether it is up or down. This is the effective service time $S_e$. The second horizontal line shows the time the job is being serviced, that is, $S_0$. The points marked by a $\times$ symbol, indicate the occurrence of failures. The distance between two consecutive $\times$-s in service time has exponential distribution (since the uptimes are exponential.) By following the reasoning that lead to eq. (1.3), we find that the number of $\times$-s, i.e., the number of failures, that occur in service time during the interval $S_0$, has a Poisson distribution with parameter $\lambda_f S_0$. Finally we see that the total failure time is $S_I$, the sum of the service interruption times.

Clearly we have to compute $\mathbb{E}(N|S_0)$, which follows now. We know that machine uptime is an exponential random variable with parameter $\lambda_f$. Hence the number of uptimes needed to provide a total amount of service $S_0$ has a Poisson distribution. In fact, if the server needed $N + 1$, say, consecutive uptimes to process a job, $N$ failures occurred, hence,

$$
\mathbb{E} \left( \sum_{i=1}^{N} D_i | S_0 \right) = \lambda_f S_0 \mathbb{E} D = \frac{m_r}{m_f} S_0.
$$

Here $\lambda_f$ stands for the failure arrival rate and equals $1/m_f$. Now the effective mean service time $E_{S_e}$ of the machine can be given by:

$$
E_{S_e} = E_{S_0} + \mathbb{E}(\mathbb{E}(N|S_0)) E_D = E_{S_0} (1 + \lambda_f m_r) = \frac{E_{S_0}}{A}. \quad (1.49)
$$

When we also want to know the effect of failures on the variability we need to determine the variance of the effective process time. When we assume that the mean time to failure and the mean time to repair are exponentially distributed we find for the variance:

$$
\sigma_e^2 = \left( \frac{\sigma_0}{A} \right)^2 + \frac{2(1 - A)}{A} E_{S_0} m_r, \quad (1.50)
$$

where $\sigma_0$ is the standard deviation of the service time without failures. There exists expressions that handle more general up and down time distribution, but here we will be satisfied with these results. The squared coefficient of variation
for service times with failures can then be given by:

$$C_e^2 = C_0^2 + 2A(1 - A) \frac{m_r}{E S_0}$$ (1.51)

with $C_0^2$ being the SCV of the service time without failures.

We will now prove the expression for the variance of $S_e$. We start by deriving a simple identity, although it may seem slightly obscure for the moment. We have, since the random variables $D_i$ are i.i.d,

$$E \left( \sum_{i=1}^{N} \sum_{j \neq i} D_i D_j \bigg| S_0 \right) = E(N(N-1)|S_0) (ED)^2$$

$$= m_r^2 \sum_{n=0}^{\infty} n(n-1) P(N = n|S_0).$$

The number of failures that arrive during a fixed interval of time has a Poisson distribution, because the failure interarrival time is assumed to be exponentially distributed. Hence,

$$E \left( \sum_{i=1}^{N} \sum_{j \neq i} D_i D_j \bigg| S_0 \right) = m_r^2 \sum_{n=0}^{\infty} n(n-1) e^{-S_0/m_f} n! \left( \frac{S_0}{m_f} \right)^n$$

$$= m_r^2 e^{-S_0/m_f} \left[ u^2 \left( \frac{\partial}{\partial u} \right)^2 \sum_{n=0}^{\infty} \frac{u^n}{n!} \right] _{u = S_0/m_f}$$

$$= \frac{m_r^2}{m_f} S_0^2.$$

From this we get for $E(S_e^2|S_0)$:

$$E(S_e^2|S_0) = E \left( S_0^2 + \sum_{i=1}^{N} D_i^2 + \sum_{i=1}^{N} \sum_{j \neq i} D_i D_j + 2S_0 \sum_{i=1}^{N} D_i \bigg| S_0 \right)$$

$$= S_0^2 + E D^2 E(N|S_0) + \frac{m_r^2}{m_f} S_0^2 + 2S_0 m_r E(N|S_0)$$

$$= E D^2 \frac{S_0}{m_f} + S_0^2 \left( 1 + \frac{m_r^2}{m_f} + 2 \frac{m_r}{m_f} \right)$$

$$= E D^2 \frac{S_0}{m_f} + \frac{S_0^2}{A^2}.$$

Since we assume that the repair times are exponential, we have:

$$ED^2 = 2 m_r^2.$$

We can finally assemble these intermediate results together with (1.49) to get
the variance of the effective service time:

\[
\sigma_e^2 = \mathbb{E}(S_e^2) - (\mathbb{E}(S_e))^2 \\
= \mathbb{E}_{S_0}(\mathbb{E}(S_e^2 | S_0)) - \frac{(\mathbb{E}(S_0))^2}{A^2} \\
= \frac{2m_r^2}{m_f} \mathbb{E}S_0 + \frac{\mathbb{E}(S_0^2) - (\mathbb{E}(S_0))^2}{A^2} \\
= \frac{\sigma_0^2}{A^2} + \frac{2m_r^2}{m_f} \mathbb{E}S_0.
\]  

(1.52)

**Example 1.4**

Occasionally the repair men are ill. The manager knows that on the average 7% of all possible working days are lost due to illness.

**Ex. 1.11.1** Is this information enough to compute \(\mathbb{E}S_e\)?

**Ex. 1.11.2** Can the manager find \(C_e^2\)? If not what other information does she need?

**Ex. 1.11.3** Is the expression for \(C_e^2\) reasonable for this situation? Hint: Is it reasonable to assume that the ‘repair’ (= time to recover from illness) time of an ill person is exponentially distributed?

Suppose the shop’s output depends critically on one machine. If it breaks down, it takes two days to have it repaired. If the arrival rate of faulty items is Poisson with arrival rate \(\lambda = 5\) per day, and the shop can contain at most 20 items,

**Ex. 1.11.4** Can you compute the probability that the shop’s capacity will be exceeded when it is empty when the machine breaks down?

**Ex. 1.11.5** What if it contains 10 items?

---

### 1.11.2 Setup Times

When a machine has to switch from processing on type of job to another, it often requires some kind of adjustment or modification, e.g. a new color of paint, or changing reels of transistors and resistors when a different type of printed circuit board is to be produced. The time involved to make this change is called ‘setup’ time. As setup is part of normal machine maintenance it is regarded as a non-preemptive outage. Here we will restrict ourselves to the single machine case.

During the setup time, a machine is obviously not productive. Therefore these setup times should be as short as possible. However, not only the average setup time has an impact on the queueing time. When it is a random variable with positive variance, we can expect from Kingman’s formula, eq. (1.31), that
variability in the setup time will negatively affect the queue size, that is, increase it. Thus we want to be able to quantify the influence of setup time on the average and SCV of the service time of jobs.

It is immediately clear that handling jobs in batches will reduce the effect of setups, because the time of a setup is spread over all jobs in the batch, instead of each job requiring an adjustment of the machine. Let us define $N_s$ as the number of jobs in one batch, i.e. the number of jobs processed between two consecutive setups, and $\mathbb{E}S_s$ as the mean setup time. Then we have for the effective service time, including setups,

$$\mathbb{E}S_e = \mathbb{E}S_0 + \frac{\mathbb{E}S_s}{N_s}, \quad (1.53)$$

When we assume that the probability of setup after each batch is constant we can find an expression for the variance of the effective service time. Writing the variance of the setup times as $\sigma_s^2$, we have

$$\sigma_e^2 = \sigma_0^2 + \frac{\sigma_s^2}{N_s} + \frac{N_s - 1}{N_s^2} (\mathbb{E}S_s)^2. \quad (1.54)$$

Formula 1.54 is derived as follows.

From $P(S_e = S_0 + S_s) = \frac{1}{N_s}$ and $P(S_e = S_o) = \frac{N_s - 1}{N_s}$ we obtain

$$\sigma_e^2 = \mathbb{E}S_0^2 + \frac{2}{N_s} \mathbb{E}S_0 \mathbb{E}S_s + \frac{1}{N_s} \mathbb{E}S_s^2 - (\mathbb{E}S_0)^2 - \frac{2}{N_s} \mathbb{E}S_0 \mathbb{E}S_s - \frac{1}{N_s^2} (\mathbb{E}S_s)^2 = \sigma_0^2 + \frac{1}{N_s} \sigma_s^2 + \frac{N_s - 1}{N_s^2} (\mathbb{E}S_s)^2.$$

Finally, the SCV for the effective service time follows immediately from

$$C_e^2 = \frac{\sigma_e^2}{(\mathbb{E}S_e)^2}. \quad (1.55)$$

**Ex. 1.11.6** What different causes of variations in setup time can you think of?

**Ex. 1.11.7** What happens if $N_s$ is allowed to vary, i.e. is a random variable itself?

### 1.11.3 Rework

As we mentioned before, rework is also a non-preemptive outage. Jobs require rework when, after having completed a machine’s service, they do not pass a quality check, but instead are offered to the station again to undergo another service cycle. When a fraction $f_r$, assumed constant and the same for all jobs, of the jobs is sent back to the queue for rework, then on the average 1 out of $1/f_r$ jobs will be sent back. (The arrivals of jobs requesting rework are geometrically distributed). Supposing that a failed job requires an amount of service $S_r$, we
establish similar equations for the mean and variance of the effective service time in the case of rework as for setups, eq. (1.53, 1.54):

\[
\mathbb{E}S_e = \mathbb{E}S_0 + \mathbb{E}S_r f_r, \quad (1.56)
\]

\[
\sigma^2_e = \sigma^2_0 + \sigma^2_r f_r + f_r (1 - f_r) (\mathbb{E}S_r)^2, \quad (1.57)
\]

Equation (1.55) applies to obtain the SCV.

1.12 Process Batching

There are two main reasons for batching jobs (a batch is a group of jobs) together in a manufacturing system. These two reasons are very different and therefore we distinguish between two kinds of batches.

1. **Avoiding setups:** From equation (1.53) it is easy to see that when the number of serviced jobs between two setups increases, the effective service time will decrease. This in turn may lead to reduced time in the system. As all jobs arriving between two consecutive setups will be processed together, this type of batch is called a *process batch*. Usually one machine will handle an entire batch, otherwise a number of servers have to be set up. Therefore the case of a single server handling batches is the most interesting one, practically speaking.

2. **Facilitating material handling:** For instance, suppose that printed circuit boards are produced in process batches of size 150, and they have to be brought to the testing department upon completion of production, it will require less work for one person to move 15 boxes of 10 printed circuit boards than to carry them all separately. This type of batch is called a *move batch* and is defined as the number of jobs moved together from one station to the next.

The latter type of batch only affects the expected travel time from one station to the next, which can clearly only happen in queueing networks. Therefore we will introduce move batches in the next chapter. Here we will discuss the effect of process batches on the service time and its SCV.

We distinguish three phases in the batch queueing process. During the first phase jobs arrive at a station, and wait to form a batch until the batch is full. When the tagged batch is complete it joins a queue of *batches*, which we assume from now on is processed by a single FIFO server. Before the processing of the tagged batch starts, it has to wait until all batches in front of it have been processed. Hence the service time of entire batches make up the waiting time of the tagged batch in queue. The last phase is the processing of the jobs that constitute the tagged batch. We summarize these phases to introduce some notation,

- \(W^B_f\) = time to form the batch,
- \(W^B_Q\) = queuing time of the batch,
- \(S_{eB}\) = effective service time of batches in front of the tagged batch
• $S_B$ = processing time of the tagged batch.

The average total time in the system is clearly

$$E W_B = EW_f^B + EW_Q^B + ES_B.$$  \hfill (1.58)

We now discuss these phases separately to derive the relevant performance measures, but first make some assumptions explicit. Job arrivals and their service requirements are i.i.d. The arrival rate is $\lambda$, the service time is $ES_0$. The setup time is independent of the arrival and service process. The station has a single machine that processes jobs according to FIFO scheduling.

For further insights of batching on the total manufacturing system, see e.g. [39].

**Forming the batch** The first measure of interest is the batch arrival rate $\lambda_B$ and relates to the job arrival rate $\lambda$. A batch of size $N$ is ready when the $N$-th job arrives. Hence we can think of batches arriving at every $N$-th job, so that

$$\lambda_B = \frac{\lambda}{N}.$$ \hfill (1.59)

The waiting time to form a batch can be easily obtained when we ‘think backwards in time’. The $N$-th job does not have to wait before the batch is full. (In fact the $N$-th job’s arrival completes the batch.) The $N-1$-th job has to wait until the $N$-th job arrives. This takes on the average $1/\lambda$. The $i$-th job, in order of arrival, should wait $\sum_{j=1}^{N} A_j$, where $A_i$ is the interarrival time between job $i-1$ and job $i$. Applying this reasoning to all jobs, we have

$$EW_f^B = \frac{1}{N} \mathbb{E} \left( \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} A_j \right) = \frac{N-1}{2\lambda}.$$ \hfill (1.60)

The last measure is the SCV of the batch arrival process, $C_{aB}$. Using the fact that the arrivals are i.i.d., we find

$$C_{aB}^2 = \lambda_B^2 \text{Var} \left( \sum_{i=1}^{N} A_i \right)$$ \hfill (1.61)

$$= \frac{\lambda^2}{N^2} N \text{Var} A_1$$ \hfill (1.62)

$$= \frac{C_a^2}{N},$$ \hfill (1.63)

where $C_a^2$ stands for the SCV of the interarrival time of one job.

**Batch queueing** As stated above, the service time of the batches in front of the tagged batch determine its waiting time in queue. Thus we need to derive the effective batch service time and its related SCV. We know that before the machine can start processing, it requires a certain set up time $S_s$, so that

$$ES_{sB} = \mathbb{E} \left( S_s + \sum_{i}^{N} S_i \right) = ES_s + N ES_0.$$ \hfill (1.64)
1.12. PROCESS BATCHING

Now we can define the workload on the machine as
\[ \rho = \lambda_B \mathbb{E}S_{eB}. \]  
(1.65)

Since we require system stability, i.e. \( \rho < 1 \), we can find a lower bound on the batch size:
\[ N > N_{\text{min}} = \frac{\lambda \mathbb{E}S_s}{1 - \lambda \mathbb{E}S_0}, \]  
(1.66)

where we used (1.64).

The SCV of the batch is, by the independence of the job service times and setup time,
\[ C_{sB}^2 = \frac{\text{Var} \, S_{eB}}{(\mathbb{E}S_{eB})^2} = \frac{\text{Var} \, S_s + N \text{Var} \, S_0}{(\mathbb{E}S_s + N \mathbb{E}S_0)^2}, \]  
(1.67)

With the SCV of the batch arrival and effective service times, we can apply Kingman’s approximation, to find
\[ \mathbb{E}W_B^Q = C_{aB}^2 + C_{sB}^2 \cdot \frac{\rho}{1 - \rho} \mathbb{E}S_{eB}, \]  
(1.68)

where of course the utilization is defined as in equation (1.65).

**Processing the tagged batch** During the last queueing phase the jobs are processed themselves. We distinguish two obvious cases here, but there are clearly more:

1. Jobs cannot leave the station by themselves, but have to wait for the entire batch to be processed.
2. After having received its service the job leaves the station.

In the first case all jobs have to wait for the last one to finish service, thus
\[ \mathbb{E}S_B = \mathbb{E}S_{eB} = \mathbb{E}S_s + N \mathbb{E}S_0. \]  
(1.69)

When jobs leave as specified by the second principle, the first job spends \( S_s + S_0 \) in service; the second \( S_s + 2S_0 \), etc. The average service time for an arbitrary job should therefore equal
\[ \mathbb{E}S_B = \mathbb{E}S_s + \frac{1}{N} \mathbb{E} \left( \sum_{i=1}^{N} \sum_{j=1}^{i} S_j \right) = \mathbb{E}S_s + \frac{N + 1}{2} \mathbb{E}S_0. \]

**Example 1.5** Suppose jobs arrive at an M/M/1 process batch queue with a rate of 10 per day, and are serviced with rate 30 per day. The machine requires a fixed set up time of 2 days.

**Ex. 1.12.1** What is the minimal batch size to make the system stable?

**Ex. 1.12.2** What is the SCV of the batch service process if the batch size is 50?
Figure 1.6: Expected waiting time at a machine as a function of process batch size

**Ex. 1.12.3** If you would have to determine the optimal batch size in this case, but the available data are not very accurate, would you advise to make the process batches larger or smaller than the computed optimum?

*Figure 1.6 shows the graph of the total waiting time as function the batch size. Note the linear increase when N becomes somewhat larger than its minimum value.*

### 1.13 Single stage production-to-stock systems

So far, we have assumed that jobs arrive at a workstation where, after a possible waiting time, operations may take place, after which one product (or a batch of products) leaves the system to fulfil customer demand. When however the type of product that will be demanded can be predicted, and when in addition customers would ideally like to be served immediately, it makes sense to produce a number of products in advance and to store them in a finished products warehouse. In this section we study a simple case in which customer demand generates to activities. Assume that we wish to hold a co-called base-stock level of $S$ finished products, to be stored in a warehouse (or a retail store). Each customer arriving is assumed to demand for exactly one product. The inventory manager checks if the product is in stock and, if so, delivers it to the customer who leaves the system immediately. If no items are in stock, the customer joins a queue of waiting customers that are served in the order of arrival (hence each case a product becomes available and there is still a backlog, it fills the demand of the longest waiting customer). Independent of whether a customer can be served immediately or not, the inventory manager immediately places an order at the manufacturing system to produce one item in order to replenish the stock. See figure 1.7 for a sketch of the procedure.
1.13. SINGLE STAGE PRODUCTION-TO-STOCK SYSTEMS

For the moment, we assume that the production system consists of a single machine that produces items with an exponential production time at rate $\mu$. Customers arrive according to a Poisson process with rate $\lambda$. We denote by $N$ the number of jobs in or waiting for production, $M$ the number of finished products still in stock, and $K$ a possible backlog (unfilled customer orders because the finished item stock is depleted). Finally, let $S$ be the base-stock or order-up-to level. Since each arriving customer demands for exactly one product and is prepared to wait in case no finished items are available, the following relations hold for $N = n, M = m$ and $K = k$

$$n + m - k = S$$
$$m.k = 0$$

One way to verify these relations is as follows. Clearly, there are no customers waiting as long as finished products in stock are available, which shows the second relation. As long as $n < S$, we have $m = S - n > 0$ and hence $k = 0$, hence in that case the first relation is fulfilled. If $n \geq S$, we have $m = 0$ and $k = n - S$, again satisfying the second relation. Note that these arguments imply that, although three variables are relevant, the variable $n$ in fact completely describes the state of the system. Typical performance measures of interest in this system concern the probability distribution of the backlog (the number of customers waiting because stock is depleted). Let as before $\rho = \frac{\lambda}{\mu}$, then we have

$$\mathbb{P}(K = 0) = \sum_{n=0}^{S} \mathbb{P}(N = n) = \sum_{n=0}^{S} (1 - \rho)\rho^n = 1 - \rho^{S+1}$$
$$\mathbb{P}(K = k) = \mathbb{P}(N = S + k) = (1 - \rho)\rho^{S+k}, \quad k > 0.$$  

Also the probability $q$ that an arriving customer has to wait, given that he is able to observe the presence and size of a queue of waiting customers, is often of interest. We have

$$q = 1,$$
$$q = \frac{\mathbb{P}(N = S)}{\mathbb{P}(N \leq S)} = \frac{(1 - \rho)\rho^{S}}{1 - \rho^{S+1}}, \quad \text{if } k = 0,$$

since a arriving customer observing other customers waiting can only place his order and join the tail of the queue. If an arriving customer observes an empty queue, there is still a probability $q$ that he has to wait, in the case that the stock has just depleted.
The above analysis is useful for determining the size of the base stock level $S$. A criterion often used is that of the fill rate $FR$, i.e. the fraction of customers of which demand can be fulfilled immediately (from stock). Due to the PASTA property, we immediately deduce

$$FR = \mathbb{P}(N \leq S - 1) = 1 - \rho^S$$

If a target fill rate of $\alpha$ is required (e.g. $\alpha = 0.95$) then $S$ is determined as the smallest value for which $1 - \rho^S \geq \alpha$.

Finally, the reader may note that similar arguments apply for the case of a more complex workstation, e.g. a multi-server system. What only counts is the fact that the number of items in production in fact determines the state of the entire system and hence also the backlog.
Chapter 2

Job Shop Manufacturing Systems

2.1 Introduction

In the previous chapter we discussed single stage manufacturing systems. In this chapter we give an introduction to Job Shops that can be modelled as networks of workstations. We start with discussing single-class systems, modelled as a network of workstations in which each workstation may consist of multiple servers. Thus, in queuing terminology we are dealing with a queueing network that is composed of \( M \) multiserver stations with \( c_i \geq 1 \), machines at station \( i \), \( (i = 1, \ldots, M) \). First, we consider the case in which all external arrival processes are Poisson (hence have exponential and mutually independent interarrival times), while also all service times are exponentially distributed. All products belong to one class, hence have the same arrival, service and routing parameters. Next, we turn to more general systems, i.e. systems in which the arrival process is a renewal process (with general interarrival time distribution) and in which also the operations at the workstations can have a general service time distribution. Finally, we briefly discuss multi-class systems, i.e. systems in which we distinguish multiple job classes with different routings and service time distributions.

2.2 Open single class product form networks

2.2.1 A Tandem Queue

In this section we introduce open queueing networks by giving a simple example of a two station flow shop. In the next subsection we will generalize this to arbitrarily structured open queueing networks.

In the two station single server flow shop of figure 2.1 jobs arrive at station 1 according to a Poisson process with rate \( \lambda \). Here they possible join a queue, are processed, and then move to station 2 for a second processing step. Upon completion of service at station 2 the jobs leave the network. We assume that service requests at stations 1 and 2 are exponentially distributed with parameters \( \mu_1 \) and \( \mu_2 \). Of course we want system stability, so that the utilization \( \rho_i \) of
CHAPTER 2. JOB SHOP MANUFACTURING SYSTEMS

Figure 2.1: Two queues in tandem

Figure 2.2: The transitions, and associated rates, between the first few states of the two queue tandem network.

station $i$ satisfies

$$\rho_i = \frac{\lambda}{\mu_i} \quad \rho_i < 1.$$  \hfill (2.1)

As in the previous chapter, in which we used state probabilities $\pi(n)$ to obtain performance measures, we need to describe the possible states of the queueing system and assign probabilities to them. Since we now have two single server stations, we need two parameters, $n_1$ and $n_2$ respectively, to represent the number of jobs at station 1 and 2, or briefly, the tuple $(n_1, n_2)$. Figure 2.2 shows part of the transitions and the rates between the states.

The next step is to use the balance equations to find the probability $\pi(n_1, n_2)$ in state $(n_1, n_2)$. Again we apply the principle that the rate of jobs out of a state should be equal to the rate into it. For the two station shop we now find:

$$\lambda \pi(0, 0) = \mu_2 \pi(0, 1),$$

$$(\lambda + \mu_1) \pi(n_1, 0) = \mu_2 \pi(n_1, 1) + \lambda \pi(n_1 - 1, 0),$$

$$(\lambda + \mu_2) \pi(0, n_2) = \mu_2 \pi(0, n_2 + 1) + \mu_1 \pi(1, n_2 - 1)$$

$$(\lambda + \mu_1 + \mu_2) \pi(n_1, n_2) = \mu_2 \pi(n_1, n_2 + 1) + \mu_1 \pi(n_1 + 1, n_2 - 1) + \lambda \pi(n_1 - 1, n_2),$$

$$\mu_1 \pi(n_1, n_2) = \mu_2 \pi(n_1, n_2 - 1) + \mu_1 \pi(n_1 + 1, n_2 - 1) + \lambda \pi(n_1 - 1, n_2).$$

(2.2)
where \( n_1, n_2 > 0 \). This time we will not try to solve these balance equations algebraically, which will require quite a lot of work, and then impose the normalization equation:

\[
\sum_{n_1, n_2} \pi(n_1, n_2) = 1
\]  

(2.3)

to scale \( \pi(0,0) \). Instead we will propose a solution—which is nothing but a smart guess—and verify that it satisfies the above balance equations. Then we will use a general result stating that the solution of any set of balance equations, if it can be found, is unique, see e.g. [67] for a proof. Hence our guess, once it can be shown to satisfy the balance equations, will be the unique solution.

To provide some intuition as to why this approach works, we first note that station 1 is just an M/M/1 queue. Hence, no matter the state of the second queue, we know that the probability that there are \( n_1 \) jobs in station 1 should be

\[
\pi(n_1) = \sum_{n_2=0}^{\infty} \pi(n_1, n_2) = (1 - \rho_1) \rho_1^{n_1}.
\]  

(2.4)

The next step is based on Burke’s theorem which states the remarkable fact that the arrival process at station 2 also being a Poisson process, the queue at station 2 should be an M/M/1 queue as well. Thus, with \( \rho_2 = \lambda/\mu_2 \),

\[
\pi(n_2) = \sum_{n_1=0}^{\infty} \pi(n_1, n_2) = (1 - \rho_2) \rho_2^{n_2}.
\]

Guided by the insight that Burke’s theorem provides, we infer that the number of jobs at station 2 is independent of the number of jobs at station 1. But then the simultaneous stationary distribution of the number of jobs at station 1 and 2 should be the product of the marginal distributions at both stations! Hence we propose as a solution

\[
\pi(n_1, n_2) = \pi(n_1) \pi(n_2) = \prod_{i=1}^{2} (1 - \rho_i) \rho_i^{n_i}.
\]  

(2.5)

Expressions of this type are known as ‘product form’ solutions, as suggested by the form.

**Ex. 2.2.3** If you know that station 1 is empty, what do you know about the arrival process at station 2?

Now that we have guessed a solution, all we have to do is substitute it in the balance equations (2.2) and verify that it indeed satisfies these equations.
Ex. 2.2.4 Check this.

Have found the stationary distribution of the number of jobs at the stations, we can determine the expected number of jobs in the system. Note that it is just the sum of \( \mathbb{E}L_1 \) and \( \mathbb{E}L_2 \).

\[
\mathbb{E}L = \sum_{n_1, n_2} (n_1 + n_2) \pi(n_1, n_2)
= \sum_{n_1} n_1 \rho_1^{n_1} (1 - \rho_1) \sum_{n_2} \rho_2^{n_2} (1 - \rho_2) + \sum_{n_1} \rho_1^{n_1} (1 - \rho_1) \sum_{n_2} n_2 \rho_2^{n_2} (1 - \rho_2)
= \frac{\rho_1}{1 - \rho_1} + \frac{\rho_2}{1 - \rho_2}.
\] (2.6)

Here we used the trivial result
\[
\sum_{n_i} \rho_i^{n_i} (1 - \rho_i) = 1, \quad i = 1, 2. \tag{2.7}
\]

From Little’s law we find for the time in the system:

\[
\mathbb{E}W = \frac{\mathbb{E}L}{\lambda}
= \frac{1}{1 - \rho_1 \mu_1} + \frac{1}{1 - \rho_2 \mu_2}.
\] (2.8)

To find the expected waiting time in the queues we need to subtract both service times from \( \mathbb{E}W \).

2.2.2 Jackson-networks

The above example of a two station flow shop can be generalized to an arbitrarily structured open queueing network with exponential processing times. The first results in the analysis of queueing networks were presented by Jackson in 1963, see e.g. [22]. He introduced what is now called the single-class product-form open queueing networks. They are defined as follows.

**Definition 2.1** A Jackson-network is a single-class open queueing network with \( M \geq 1 \) stations and with the following characteristics. Station \( i \) has \( c_i \geq 1 \) servers and the service times are exponentially distributed with parameter \( \mu_i > 0 \). The service discipline employed is first-come first-served at all stations. The jobs arrive from outside the network at station \( i \) according to a Poisson process with intensity \( \gamma_i \). The jobs have a Markovian routing, characterized by an irreducible routing matrix \( P \).

First let us explain the Markovian routing. When a job is finished at station \( i \) it will either go to station \( j \) with probability \( P_{ij} \), or leave the network with probability \( P_{i0} = 1 - \sum_{j \neq 0} P_{ij} \). These probabilities \( P_{ij} \) form a routing matrix \( P \).

For Jackson-networks we find the fundamental traffic-rate equations:

\[
\lambda_i = \gamma_i + \sum_{j=1}^{M} \lambda_j P_{ji}, \quad i = 1, \ldots, M,
\] (2.9)
where \( \lambda_i \) denotes the total arrival rate at station \( i \). In this equation, \( \gamma_i \) denotes the arrival rate of jobs from outside the network at station \( i \) and \( \sum_j \lambda_j P_{ij} \) the arrival rate of jobs originating from other stations inside the network. We can write eq.(2.9) more succinctly in the form

\[
\bar{\lambda} = \bar{\gamma} + \bar{\lambda} \mathbf{P}
\]

(2.10)

with \( \bar{\lambda} \) and \( \bar{\gamma} \) being \( 1 \times M \) vectors, and \( \mathbf{P} \) an \( M \times M \) matrix.

It is not immediately clear that a \( \bar{\lambda} \) exists that solves (2.10). We will prove this now in the next few exercises.

**Ex. 2.2.5** Defining \( P^n = \mathbf{P} P^{n-1} \), show that \( \sum_j (P^M)_{ij} < 1 \), for all \( i \), if for at least one \( i \), \( \sum_j P_{ij} < 1 \). Interpret this result. (Hint: why do we use \( P^M \), instead of \( P \)?)

**Ex. 2.2.6** Show that eq.(2.10) can be solved for \( \bar{\lambda} \).

We also introduce the so called visit ratio’s \( V_i \), \( i = 1, \ldots, M \). These visit ratio’s satisfy the homogeneous equation \( \bar{V} = \bar{V} \mathbf{P} \), where \( \bar{V} = (V_1, \ldots, V_M) \), and are determined up to a multiplicative constant. For open queueing networks the visit ratio’s are defined as:

\[
V_i = \frac{\lambda_i}{\gamma}, \quad i = 1, \ldots, M,
\]

(2.11)

where \( \gamma = \sum_i \gamma_i \).

It can be proved that the stationary distribution has a product-form solution. Here we give the solution for a Jackson network with \( M \) stations, each having \( c_i \) servers. As before this solution is proved by substituting it into the balance equations describing the Markov process of the open queueing networks. The stationary distribution is given by:

\[
\pi (n_1, \ldots, n_M) = \prod_{i=1}^{M} f_i (n_i),
\]

(2.12)

with:

\[
f_i (n_i) = \frac{1}{G (i)} \frac{(c_i \rho_i)^{n_i}}{n_i!}, \quad n_i < c_i
\]

(2.13)

and

\[
f_i (n_i) = \frac{1}{G (i)} \frac{c_i^{n_i} \rho_i^{n_i}}{c_i!}, \quad n_i \geq c_i
\]

(2.14)

where \( \rho_i \) is the utilization of station \( i \):

\[
\rho_i = \frac{\lambda_i}{c_i \mu_i}, \quad \rho_i < 1.
\]

(2.15)

The normalization constant \( G (i) \) is defined as:

\[
G (i) = \sum_{n=0}^{c_i-1} \frac{(c_i \rho_i)^n}{n!} + \frac{(c_i \rho_i)^{c_i}}{c_i!} (1 - \rho_i)^{-1}.
\]

(2.16)

This product form solution of the stationary distribution only applies when \( \rho_i < 1 \) for all \( i \), otherwise the network would not be stable. Now we present two examples.
Ex. 2.2.7 Consider a two station network with rework. Jobs arrive only at the first station, with rate $\gamma$. A fraction $\alpha$ of served jobs at station 1 is fed back to itself, the rest, $1 - \alpha$, is sent directly to station 2. There a fraction $\beta_2$ is sent to itself, $\beta_1$ to station 1, and $1 - \beta_1 - \beta_2$ leaves the network. Write down the traffic rate equations and determine the overall arrival rate for each station. Formulate appropriate stability requirements on $\alpha$, $\beta_1$ and $\beta_2$. What happens if you let $\beta_1 = 0$, or $\alpha = 1$?

Ex. 2.2.8 Is the arrival process at station 1 still Poisson when $0 < \alpha < 1$?

2.2.3 Performance measures for Jackson networks

We are interested in the performance measures of each individual station and for the network as a whole. For instance, when looking at individual stations we might identify a bottleneck station, that is, a station having much longer queues than the other stations. From equations (2.12) through (2.16) we can see that the probability of having $n$ jobs at station $i$ is independent of the state of all other stations in the network. Therefore we may compute the performance measures for individual stations separately and then add them to obtain the measures for the whole network.

For the throughput we define:

$$TH = \gamma = \sum_{i=1}^{M} \gamma_i.$$  \hfill (2.17)

The expected number of jobs at station $i$ is, see (1.41)

$$E L_i = \frac{(c_i \rho_i)^{c_i}}{c_i! G(i)} \frac{\rho_i}{\left(1 - \rho_i\right)^2} + c_i \rho_i.$$ \hfill (2.18)

Again we can apply Little’s law to get an expression for $E W_i$. The expected total number of jobs in the entire network is now easy:

$$EL = \sum_{i=1}^{M} EL_i.$$ \hfill (2.19)

When computing the expected time in the system we have to take the visit ratios as obtained from eq. (2.9) and (2.11) into account. Hence the expected time in the system of a job is

$$EW = \sum_{i=1}^{M} V_i EW_i = \sum_{i=1}^{M} \frac{EL_i}{\lambda_i}.$$ \hfill (2.20)

Note that

$$EL = \sum_{i=1}^{M} EL_i = \sum_{i=1}^{M} \lambda_i EW_i = \sum_{i=1}^{M} \gamma V_i EW_i = \gamma EW,$$

which is Little’s law applied at the system level.
2.2. OPEN SINGLE CLASS PRODUCT FORM NETWORKS

2.2.4 Multiple class Jackson Networks

So far, we have discussed the case in which all jobs can be characterized as probabilistically the same, i.e. they have the same arrival and service parameters and the same probabilistic routing mechanism. In fact, jobs may be different but can we assume they can be aggregated into a single class. There are however a number of cases in which such a rough aggregation is not acceptable, for instance because the routing (machine/operation sequence) and the processing requirements of different classes of jobs are vastly different. In this section, we concentrate on situations in which the operation/machine sequence is different for different classes of jobs, but the service requirements of the different classes at any workstation are probabilistically almost the same. In addition, we allow that jobs may change their class. This helps to model situations in which jobs may visit a machine more than once but the routing mechanism after the first and second visit are vastly different. In such a case, after completing the first visit to a particular workstation, the job changes its class to distinguish it from jobs that still have to pay their first visit to that workstation. It turns out that for such systems, a product form solution still applies.

Assume that we have $M$ workstations and $R$ job classes. Similar to the preceding section, we denote by $\gamma_i^{(r)}$ the external arrival rate of class $r$ jobs to workstation $i$. Let $P_{ij}^{(s,r)}$ denote the probability that a class $r$ jobs, after leaving workstation $i$, becomes a class $s$ job and moves to workstation $j$. The traffic-rate equations now become

$$\lambda_i^{(r)} = \gamma_i^{(r)} + \sum_{j=1}^{M} \sum_{s=1}^{R} \lambda_j^{(s)} P_{ji}^{(s,r)} , \quad i = 1, \ldots, M; r = 1, \ldots, R, \quad (2.21)$$

As noted earlier, we assume that all jobs have an exponential service time at workstation $i$ with parameter $\mu_i$, independent of their job class. Now, let $\lambda_i = \sum_{r=1}^{R} \lambda_i^{(r)}$ indicate the overall arrival rate at workstation $i$, and let the workstation-dependent parameters $\rho_i, G(i)$ and $f_i(n_i)$ be defined by equations 2.15, 2.16, 2.13 and 2.14. Then it can be shown that the joint probability distribution of all job classes at all workstations satisfies

$$\pi(n_1^{(1)}, n_2^{(1)}, \ldots, n_1^{(R)} , n_2^{(R)}, \ldots, n_M^{(1)}, n_M^{(2)}, \ldots n_M^{(R)}) = \prod_{i=1}^{M} \binom{n_i}{n_i^{(1)}, \ldots, n_i^{(R)}} \prod_{r=1}^{R} \left( \frac{\lambda_i^{(r)}}{\lambda_i} \right)^{n_i^{(r)}} f_i(n_i) \quad (2.22)$$

From this probability distribution, again more general performance measures per class can be derived. We will come back to this when discussing more general, non-product-form networks. In particular the case in which we deal with multiple job classes where the service rates are still exponential but also class-dependent, a simple product form solution no longer applies, let alone in cases where service times are no longer exponential at all. These topics are discussed in the next two sections.
2.3 Production to stock models

Similar to the single stage case, we may encounter systems in which a certain number of products are produced in anticipation of future demand, and held in stock until demand materializes. This typically occurs for commodities in the consumer market that are produced in large quantities and with limited variation. We start with a simple product form network model in which only one class of products is produced. Demand arrives according to a Poisson process with rate $\lambda$. In addition, a base stock level of $S$ items is the driver for production. Hence, each time a customer arrives (and is assumed to request exactly one item), a product is delivered immediately from stock, if available, otherwise the customer will wait until an item becomes available from production to fulfill its needs (waiting customers are assumed to be served according to a first-come, first served discipline). In any case, upon arrival of the customer, a production order is issued to replenish the (immediately or later) delivered item. Let, as before, $\vec{N} = (N_1, N_2, ..., N_M)$ be the random vector denoting the number of items at each workstation, $M$ the number of items in stock, and $K$ the backlog (or equivalently, the number of customers waiting for a product). Figure 2.3 displays a single class production to stock manufacturing network, with $\vec{N} = \vec{n}, M = m$, and $K = k$.

Clearly, the number of items in production is now described by the vector $\vec{n} = (n_1, n_2, ..., n_M)$, denoting the number of items at each workstation. Similar to the single stage system, we may easily verify the following relations:

$$|\vec{n}| + m - k = S$$
$$m.k = 0$$

Indeed, as long as $|\vec{n}| \leq S$ we have that $m = S - |\vec{n}|$ and $k = 0$, while $|\vec{n}| > S$ implies $m = 0$ and $k = |\vec{n}| - S$. Hence, since we deal with a product form network, we find for the distribution of the backlog $K$

$$P(K = 0) = P(|\vec{N}| \leq S) = \sum_{n=0}^{S} \sum_{n_1, ..., n_M: \sum_{i=1}^{M} n_i = n} \prod_{i=1}^{M} P(N_i = n_i)$$

$$P(K = k) = P(|\vec{N}| = S + k) = \sum_{n, ..., n_M: \sum_{i=1}^{M} n_i = S + k} \prod_{i=1}^{M} P(N_i = n_i), \quad \text{for } k > 0$$
2.4. GENERAL JOB SHOP MANUFACTURING SYSTEMS

Also, the fill rate $FR$ follows immediately from the PASTA property, hence

$$\text{FR} = \mathbb{P}(N \leq S - 1)$$

It is not hard to extend the above analysis to multi-class product form manufacturing networks. In this case, an inventory and hence a base stock level $S^{(r)}$ is defined for each class $r, r = 1, 2, ..., R$. Therefore, we also end up with $R$ different stock points and $R$ arrival streams, leading to $R$ synchronization queues of the type displayed in Figure 2.3. For these systems, it generally does not make sense to allow for the possibility that an item changes its class during production. This simplifies in particular the traffic rate equations (2.21) that now are decomposed into $R$ isolated sets of equations (one for each class). The further analysis of multi-class systems is left to the reader.

2.4 General Job Shop Manufacturing Systems

Real-world manufacturing systems seldomly obey the exponentiality assumptions that form the basis for the product form solutions discussed above. If we relax the assumption regarding exponential interarrival time and service time distributions, we can model these systems as networks of $G/G/c$ queues instead of the formerly used $M/M/c$ queues. In the new case the exact decomposition into $M$ different queues, based on the product form solution, is no longer valid. In fact there are no known closed form expressions for the performance measures for $G/G/c$ networks. Therefore, to be able to obtain approximate performance measures, we will simply assume that we are still allowed to decompose the network into $M$ individual $G/G/c$ queues, and then link these $G/G/c$ queues. Thus the performance measures will be the same as in Section 2.2.3, with the difference that the equations for the $G/G/c$ queue determine the expected waiting time. As before, we start our discussion with single class manufacturing systems.

2.4.1 Single Class Manufacturing Systems

As we now deal with a network of $G/G/c$ queues we need some information beyond what is already required for $M/M/c$ queues. Since the SCV of the interarrival and service times of jobs arriving at any workstation, $C^2_a$ and $C^2_s$ respectively, do not have to be equal to 1 anymore, we have to determine these for the expected waiting time in a $G/G/c$ queue, see eq. (1.46). The SCV of the service times can be measured directly, as well as the interarrival times of external arrivals. This is, however, not true for the SCV of all interarrival times at each station as many arrivals will be departures from other stations in the network. The so-called traffic variability equations are needed to obtain the $C^2_a$ for each station based on external and internal arrivals. We will introduce these equations in the next section.

For the sake of completeness, we summarize here the input parameters to specify an open loop manufacturing system: the mean and SCV of the service times $\mathbb{E}S_i, C^2_s$, the number of servers $c_i$ of station $i$, the arrival rate $\lambda_{0i}$ (formerly denoted by $\gamma_i$) and SCV $C^2_{0i}$ of external jobs entering the network at station $i$, and, finally, the routing matrix $P$. Hence we need $5M + M^2$ input parameters to characterize an $M$ station network of $G/G/c$ queues.
In the next section we will introduce the traffic variability equations to determine the $C_a^2$. In Section 2.4.4 we will generalize the single class open loop manufacturing system presented here to multi-class open loop manufacturing systems.

2.4.2 Traffic Variability Equations

The performance measures of a single G/G/c queue depend on the job interarrival times and service requirements, and the related SCVs. In this section we will generalize the single station equations to networks of G/G/c queues. As mentioned before, the interarrival and service distributions of flows entering the network are supposed to be well specified. The real problem in the generalization is how the stations interchange jobs among themselves, that is, how the service characteristics of one station affect the interarrival processes at other stations. We will discuss this process here in general terms. In the next sections we will look at each step more closely.

In the first place we know that the departure rate of a station in equilibrium should equal its arrival rate. Hence we can always find an expression for the departure rate if we have information about the arrival rate. These we have by means of the same traffic rate equation as we derived for Jackson networks, eq. 2.9.

The SCV of the arrival process at a station is considerably more difficult to obtain. We first notice that jobs leaving a station are routed to other stations or leave the network. Thus the departure process is split into at most $M + 1$ flows: $M$ flows to all stations in the network and one flow leaving the network. (Note that a station can send traffic back to itself.) Secondly, by the same reasoning, at most $M + 1$ flows arrive at any station thereby making up the arrival process of station $j$. Thus we observe that the departure "stream" of a station is split into single job flows, and that these single flows again join to form the arrival process at other stations. The SCV of the arrival process will therefore depend on three factors:

1. Depart: The SCV of the departure process of the aggregate traffic stream out of a station.
2. Split: The SCV of a single job flow as part of the aggregate departure stream.
3. Merge: The SCV of the aggregate arrival process as a superposition of single flows.

Because of this process, we speak of 'split and merge' configurations. We will discuss the relevant equations in the next sections and conclude with a synthesis of these three steps.

Departures

The departure process depends on both the arrival process and the service distribution. We take these aspects into account as follows. When the utilization is high the station is often busy so that the interdeparture times will resemble the service times. We would then also expect that the SCV of the departure
process is similar to the SCV of the service times. On the other hand, when the utilization is low the queue will be empty most of the time. Thus the interdeparture times will be closely approximated by the interarrival times. Hence in this case the SCV of the departure process is like the SCV of the arrival process.

For a single server station, with index \(i\), say, a reasonably accurate and simple way to interpolate between the two extremes mentioned, is given by:

\[
C_{di}^2 = (1 - \rho_i^2) C_{ai}^2 + \rho_i^2 C_{si}^2 \quad i = 1, \ldots, M,
\]

(2.23)

where \(\rho_i\) as before denotes the utilization of workstation \(i\), hence \(\rho_i = \lambda_i \mathbb{E} S_i\). From this it is easy to see that when \(\rho_i \to 0\), then \(C_{di}^2 \to C_{ai}^2\) and that when \(\rho_i \to 1\), then \(C_{di}^2 \to C_{si}^2\). Clearly, the SCV of the departure process is a convex combination of the SCV of the arrival process and the SCV of the service times.

For multiple server stations several approximations have been suggested by Buzacott & Shanthikumar [22]. In this case we have \(\rho_i = \frac{\lambda_i \mathbb{E} S_i}{c_i}\). Here we present the approximation of Whitt [84]:

\[
C_{di}^2 = 1 + (1 - \rho_i^2) (C_{ai}^2 - 1) + \frac{\rho_i^2}{c_i} (C_{si}^2 - 1) \quad i = 1, \ldots, M.
\]

(2.24)

Note that for single server stations equation (2.24) reduces to equation (2.23). A drawback of this formula is that when the service times are deterministic (i.e. \(C_{si}^2 = 0\)) the actual variability in a network is higher than eq. (2.24) suggest. Therefore, it has been suggested to set \(C_{si}^2\) to \(\max(C_{si}^2, 0.2)\) to avoid that \(C_{di}^2\) will become too small.

A useful property of equations (2.23) and (2.24) is that they are exact for M/M/c queues.

**Splitting**

The departure process is split into \(M + 1\) flows. According to the Markovian routing matrix \(P\) the job goes from station \(i\) to station \(j\) with probability \(P_{ij}\), and with \(P_{i0} = 1 - \sum_j P_{ij}\) it will leave the network.

Without proof we state that for a renewal process the SCV of the flow from station \(i\) to station \(j\) equals

\[
C_{ij}^2 = P_{ij} C_{ai}^2 + 1 - P_{ij}, \quad i, j = 1, \ldots, M
\]

(2.25)

which is exact when the departure process is renewal and the probabilities \(P_{ij}\) represent independent events (e.g. Markovian routing). Note that equation (2.25) is exact for the M/M/c queue.

**Superposition**

We are interested in the SCV of the arrival process \(C_{aj}^2\) at station \(j\) which is the superposition of \(M + 1\) flows with arrival rate \(\lambda_{ij}\) and SCV \(C_{ij}^2\), \(i = 0, \ldots, M\). Here \(\lambda_{ij} = \lambda_i P_{ij}\) for \(i = 1, \ldots, M\), and \(\lambda_0\) and \(C_{0j}^2\) are input parameters. Albin [2] and Whitt [83] have done extensive research to find approximations for the superposition of flows. Whitt [84] has further simplified the approximations whose result we present here. Again the equations are exact for M/M/c queues.
The SCV of the arrival process is approximated by:

\[ C_{2}^{a_{j}} = w_{j} \sum_{i=0}^{M} Q_{ij} C_{2}^{ij} + 1 - w_{j} \quad j = 1, \ldots, M, \] (2.26)

where

\[ w_{j} = [1 + 4 \left(1 - \rho_{j}\right)^{2} (v_{j} - 1)]^{-1} \quad j = 1, \ldots, M, \] (2.27)

and

\[ v_{j} = \left[ \sum_{i=0}^{M} Q_{ij}^{2} \right]^{-1} \quad j = 1, \ldots, M. \] (2.28)

In these equations \( Q_{ij} \) denotes the proportion of the arrival flow of station \( j \) originating from station \( i \), and is given by:

\[ Q_{ij} = \frac{\lambda_{ij}}{\sum_{i=0}^{M} \lambda_{i}} \quad i = 0, \ldots, M, j = 1, \ldots, M \] (2.29)

### Synthesis

The equations for departures, splitting and superposition are all linear equations for the SCV so that we can find a set of linear equations, they are of the form:

\[ C_{2}^{a_{j}} = a_{j} + \sum_{i=1}^{M} C_{2}^{a_{i} b_{ij}}, \quad j = 1, \ldots, M, \] (2.30)

where \( a_{j} \) and \( b_{ij} \) are constants depending on the input data:

\[ a_{j} = 1 + w_{j} \left[ (Q_{0j} C_{2}^{0j} - 1) + \sum_{i=1}^{M} Q_{ij} \left[ (1 - P_{ij}) + P_{ij} \rho_{i}^{2} x_{i} \right] \right], \] (2.31)

\[ b_{ij} = w_{j} P_{ij} Q_{ij} (1 - \rho_{i}^{2}), \] (2.32)

and \( x_{i} \) is:

\[ x_{i} = 1 + c_{i}^{-0.5} \left( \max \left[ C_{2}^{0i}, 0.2 \right] - 1 \right). \] (2.33)

\( w_{j}, v_{j} \) and \( Q_{ij} \) are given by equations (2.27), (2.28) and (2.29) respectively.

### 2.4.3 Performance measures for general single class manufacturing systems

As in the case of Jackson networks, it is now relatively easy to derive general performance measures for open manufacturing systems. The main difference consists of applying the relevant formulas for the \( G/G/c \)-queues, instead of using the product form solutions as a basis. For the sake of completeness, we present the relevant formula here. Let \( \gamma = \sum_{i=1}^{M} \lambda_{0i} \) and define \( V_{i} = \frac{\lambda_{0i}}{\beta_{i}} \).

The expected time spent waiting in the queue at workstation \( i \) follows immediately from equations (1.46) and (1.44):

\[ \mathbb{E}W_{Qi} = \frac{C_{2}^{a_{i}} + C_{2}^{s_{i}}}{2} \frac{\rho_{i} \left( \sqrt{2(c_{i} + 1)} - 1 \right)}{c_{i} (1 - \rho_{i})} \mathbb{E}S_{i}, \quad \rho_{i} < 1, \] (2.34)
and from this
\[\mathbb{E}W_i = \mathbb{E}W_{Qi} + \mathbb{E}S_i.\] (2.35)

The overall expected time in the system is now determined by
\[\mathbb{E}W = \sum_{i=1}^{M} V_i \mathbb{E}W_i\] (2.36)

Note that \(\mathbb{E}L_i = \lambda_i \mathbb{E}W_i\) and \(\mathbb{E}L = \sum_{i=1}^{M} \mathbb{E}L_i\). Indeed this yields also Little’s result at the network level, i.e.
\[\mathbb{E}L = \sum_{i=1}^{M} \mathbb{E}L_i = \sum_{i=1}^{M} \lambda_i \mathbb{E}W_i = \sum_{i=1}^{M} \gamma V_i \mathbb{E}W_i = \gamma \mathbb{E}W,\]

### 2.4.4 Multi-Class Manufacturing Systems

With the equations derived until now we can approximate the performance measures of single class OQNs with generally distributed interarrival and service times. But in real-world manufacturing systems it can also occur that there are different types of jobs. We then speak of multi-class manufacturing systems. There are two ways to approximate the performance measures of multi-class OQNs: the complete reduction method and the decomposition method. The decomposition method was first introduced by Bitran & Tirupati [12]. In this section we will only discuss the complete reduction method.

In multi-class OQNs we have \(R\) job classes with general individual interarrival and service time distributions, characterized by \(\lambda_{0j}^{(r)}\) and \((C_{0j}^{(r)})^2\), and \(E_{S_j}^{(r)}\) and \((C_{S_j}^{(r)})^2\) respectively, and routing matrices \(P_{ij}^{(r)}\), with \(i, j = 1, \ldots, M\) and \(r = 1, \ldots, R\) for all parameters. The complete reduction method comprises three steps:

1. Reduction of the given \(R\) class OQN to a single class OQN by aggregating the \(R\) classes.

2. Analysis of the single class OQN.

3. Disaggregation to obtain the performance measures per class for the given \(R\) class OQN.

Step 1 reduces the \((4M + M^2) R + M\) input parameters to \(5M + M^2\) parameters. The aggregate first and second moment of the service time at station \(j\) are given by the weighted average of the service times of the individual job classes:

\[\mathbb{E}S_j = \frac{1}{\lambda_j} \sum_{r=1}^{R} \lambda_j^{(r)} \mathbb{E}S_j^{(r)}, \quad j = 1, \ldots, M,\] (2.37)
\[\mathbb{E}(S_j)^2 = \frac{1}{\lambda_j} \sum_{r=1}^{R} \lambda_j^{(r)} \mathbb{E}(S_j^{(r)})^2, \quad j = 1, \ldots, M,\] (2.38)
where $\lambda_j^{(r)}$ is the arrival rate of class $r$ jobs to station $j$:

$$\lambda_j^{(r)} = \lambda_{0j}^{(r)} + \sum_{i=1}^{M} \lambda_i^{(r)} P_{ij}^{(r)}, \quad r = 1, \ldots, R,$$

and $\lambda_j = \sum_{r=1}^{R} \lambda_j^{(r)}$ is the aggregate arrival rate of jobs to station $j$.

With the aggregate first and second moment of the service time we find for the aggregate SCV of the service times

$$C_{sj}^2 = \frac{1}{\lambda_j (\mathbb{E}S_j)^2} \sum_{r=1}^{R} \lambda_j^{(r)} \left( \mathbb{E}S_j^{(r)} \right)^2 \left( (C_{sj}^{(r)})^2 + 1 \right) - 1, \quad j = 1, \ldots, M. \quad (2.39)$$

**Ex. 2.4.1 Check this.**

The aggregate arrival rate of jobs outside the network to station $j$ is easy, $\lambda_0j = \sum_{r=1}^{R} \lambda_{0j}^{(r)}$. To find the SCV of the aggregate arrival process we can use the equations for the superposition of flows as given in Section 2.4.2, with the difference that here we take the superposition of the $R$ job flows. The example below illustrates this process.

This leaves us to find the aggregate routing probabilities. The routing probabilities of the aggregate flow, which together form the aggregate routing matrix $P$, are given by

$$P_{ij} = \frac{1}{\lambda_i} \sum_{r=1}^{R} \lambda_i^{(r)} P_{ij}^{(r)}, \quad i, j = 1, \ldots, M. \quad (2.40)$$

The input is now reduced to $5M + M^2$ input parameters and the performance measures of the aggregate job can be approximated as in a single class OQN (Step 2).

It remains to disaggregate the performance measures to obtain those per class (Step 3). We find the mean number of jobs $\mathbb{E}L_{Q,j}$ in the queue of station $j$ with

$$\mathbb{E}L_{Q,j} = \mathbb{E}L_j - \rho_j.$$

Since the time spent in the queue of station $j$ is equal for each class, we obtain

$$\mathbb{E}L_{Q,j}^{(r)} = \frac{\lambda_j^{(r)}}{\lambda_j} \mathbb{E}L_{Q,j}. \quad (2.41)$$

Furthermore, we know that the mean number of class $r$ jobs in service at station $j$ is given by $\rho_j^{(r)} = \lambda_j^{(r)} \mathbb{E}S_j^{(r)}/c_j$ and therefore we find

$$\mathbb{E}L_j^{(r)} = \mathbb{E}L_{Q,j}^{(r)} + \rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\lambda_j} \mathbb{E}L_{Q,j} + \frac{\lambda_j^{(r)} \mathbb{E}S_j^{(r)}}{c_j}. \quad (2.42)$$

By adding the number of jobs per station, we find the number of jobs in the network $\mathbb{E}L^{(r)} = \sum_{r=1}^{R} \mathbb{E}L_j^{(r)}$.

Note that if, at each station, the service times are equal for all classes, then we have the simpler expression

$$\mathbb{E}L_j^{(r)} = \frac{\lambda_j^{(r)}}{\lambda_j} \mathbb{E}L_j. \quad (2.43)$$
2.4. GENERAL JOB SHOP MANUFACTURING SYSTEMS

The expected time in the system for jobs in class \( r \) can then be found by

\[
EW^{(r)} = \sum_{j=1}^{M} V_j^{(r)} EW_j^{(r)} = \sum_{j=1}^{M} V_j^{(r)} E\left(\frac{L_j^{(r)}}{\lambda_j^{(r)}}\right),
\]

where \( V_j^{(r)} = \lambda_j^{(r)} / \gamma_j^{(r)} \) and \( \gamma^{(r)} = \sum_j \lambda_j^{(r)} \). This completes the disaggregation step, and thereby the method.

Note that equation (2.44) coincides with Little's law applied at system level

\[
g^{(r)} EW^{(r)} = \sum_{j=1}^{M} \lambda_j^{(r)} EW_j^{(r)} = \sum_{j=1}^{M} E\left(\frac{L_j^{(r)}}{\lambda_j^{(r)}}\right) = EL^{(r)}
\]

**Example 2.2** Consider a two station tandem network that serves two job types, \( A \) and \( B \). We will, in this example, compute the expected waiting time in queue assuming that jobs arrive with rates \( \lambda_A \) and \( \lambda_B \). Furthermore, we want to make a trade off between the two classes such that \( EW_Q \) is minimal as function of \( \lambda_A \) and \( \lambda_B \) such that \( \lambda_A + \lambda_B \) is constant, \( \lambda \) say.

Before we set out the computation of \( EW_Q \) in detail we remark that its value is the same for both job classes (why?) and equals the sum of the waiting time at the first station and at the second, i.e., \( EW_Q = EW_{Q,1} + EW_{Q,2} \).

The following steps are involved in computing \( EW_Q \) as a function of \( \lambda_A \) and \( \lambda_B = \lambda - \lambda_A \).

1. Define the utilization at station \( i = 1, 2 \) as

\[
\rho_i = \rho_{A,i} + \rho_{B,i};
\]

\[
\rho_{A,i} = \lambda_A E\left(S_{A,i}\right) \quad \text{and} \quad \rho_{B,i} = \lambda_B E\left(S_{B,i}\right).
\]

2. Given, at station 1, the utilization, and the rate and SCV of the arrivals of the job types, we can establish the SCV for the aggregate arrival process by the set of equations 2.29, 2.28, 2.27, 2.26 (In the order given.).

3. The expectation and SCV of the aggregate service distribution can be found with eq. 2.37 and 2.39.

4. Now we have the parameters to compute \( EW_{Q,1} \) by eq. 1.31.

5. The next step is to find the expectation and SCV of the interarrival times at the second station. Since station 2’s arrivals are the departures at station 1, we can use eq. 2.23 to find the SCV of the arrivals at station 2. Here we used \( \rho_i \) as the utilization.

6. The computation of \( EW_{Q,2} \) can proceed along the same lines as for \( EW_{Q,1} \). Note that the average interarrival times should be \( \lambda_A \) and \( \lambda_B \), respectively.

Figure 2.4 shows the dependency of \( EW_Q \), \( EW_{Q,1} \) and \( EW_{Q,2} \) on \( \lambda_A \) and \( \lambda_B \) such that the total arrival rate remains constant. The parameters were as
follows. At station 1:

\[ 20 < \lambda_A < 6.5 \quad \lambda_B = 6.5 - \lambda_A \]
\[ \mathbb{E}S_{A,1} = 1/7 \quad \mathbb{E}S_{B,1} = 1/8 \]
\[ C_{a,A,1}^2 = 1 \quad C_{a,B,1}^2 = 1 \]
\[ C_{s,A,1}^2 = 1 \quad C_{s,B,1}^2 = 1 \]

At station 2:

\[ 2\mathbb{E}S_{A,2} = 1/8 \quad \mathbb{E}S_{B,2} = 1/8 \]
\[ C_{s,A,2}^2 = 1 \quad C_{s,B,2}^2 = 5 \]

Ex. 2.4.2 It appears that \( \mathbb{E}W_{Q,2} \) decreases linearly in \( \lambda_A \) whereas \( \mathbb{E}W_{Q,1} \) increases faster than linearly. Why?

2.5 Move batches in general flow lines

As we have stated in Section 1.12 move batches are used to facilitate material handling in manufacturing systems and are defined as the number of jobs moved together from station \( i \) to station \( j \). In this section we will discuss the influence of move batches on the time it takes from completion of service at station \( i \) and the start of service at station \( j \), for an individual job. Although in principle the technique can be applied in general manufacturing job shops, we restrict ourselves in this section to flow lines, i.e. systems in which all stations are visited exactly once, and in the same order by all jobs.
First let us take a look at the physical process in a manufacturing system. Upon completion of service at station $i$ a job will exit the server to join a move batch, of size $N$. When complete, the batch will be moved to some station $j$. At station $j$ the batch will wait in a queue until all batches in front of it have been serviced (assuming FCFS). When the batch can be serviced the individual jobs will be taken into service one at a time. Thus we can distinguish four separate stages that each take up a certain amount of time:

1. Waiting time until batch completion
2. Transportation time
3. Waiting time in the queue
4. Waiting time in batch until start of service

We will discuss these steps consecutively.

**Waiting time until batch completion** When a job exits station $i$ it has to wait until the move batch is complete. In Section (1.12) we have seen that when the arrival rate equals $\lambda_i$ the average time between a job’s joining a batch and the batch completion equals:

$$E_{WB,F,i} = \frac{N - 1}{2} \frac{1}{\lambda_i}.$$  

This equation also applies for forming a move batch as the departure rate of a station equals the arrival rate, in a stable system. We also could have found this equation by seeing that the job which completes the batch does not have to wait and that the first job of the batch has to wait for $N - 1$ jobs, which arrive with intervals $1/\lambda_i$.

**Ex. 2.5.1** Why can we just take the average waiting time of the first and the last job?

**Transportation time** For now we assume that the expected transportation time $E_{T_{ij}}$ from station $i$ to station $j$ is independent of the batch size.

**Waiting time in the queue** The waiting time in a queue for a move batch is independent of the batch size and thus equal to the waiting time of an individual job. Contrary to process batches there are no setups and thus is the service time independent of the batch size. Let $\lambda$ be the arrival rate of single jobs, $E_S$ the mean service and $N$ the size of a move batch. The utilization of the station is then

$$\rho = \text{batch arrival rate} \times \text{batch service requirement}$$

$$= \frac{\lambda}{N} \frac{N E_S}{c} = \frac{\lambda E_S}{c},$$

which shows that the load is independent of the size of a move batch. From equation (1.63) we see that when the SCV of individual jobs of the arrival
process is given by \( C_a^2 \), the SCV of the batch arrival process is given by \( C_a^2 / N \). The same applies to the SCV of the service times. Hence,

\[
\begin{align*}
\mathbb{E}W_Q^B &= \frac{C_a^2 / N + C_s^2 / N \rho \left(\sqrt{2(c+1)}-1\right)}{c(1-\rho)} N \bar{E}S \\
&= \frac{C_a^2 + C_s^2 \rho \left(\sqrt{2(c+1)}-1\right)}{2 c(1-\rho)} \bar{E}S,
\end{align*}
\]

which shows our claim.

**Waiting time in batch until start of service** The first job does not have to wait to be taken into service at station \( j \), but the last job has to wait for \( N - 1 \) jobs. The jobs can be taken into service at station \( j \) with time intervals equal to \( \bar{E}S_j / c_j \) (and not \( 1/\lambda_j \)), thus the waiting time until start of service is

\[
\mathbb{E}W_{E,j}^B = \frac{N - 1 \bar{E}S_j}{2 c_j} \tag{2.46}
\]

**Result** Combining the four stages gives the time interval between completion of service at station \( i \) and the start of service at station \( j \) for an arbitrary job

\[
\begin{align*}
\mathbb{E}W_{E,i}^B &+ \mathbb{E}T_{ij} + \mathbb{E}W_{Q,j}^B + \mathbb{E}W_{E,j}^B \\
&= \frac{N - 1}{2} \left( \frac{1}{\lambda_i} + \frac{\bar{E}S_j}{c_j} \right) + \mathbb{E}T_{ij} + \mathbb{E}W_{Q,j}^B. \tag{2.47}
\end{align*}
\]

From equation (2.47) our first impression would be that the optimal batch size is always 1, but this is not so. Note that we have assumed that the transportation time is independent of the batch size, but if we model the transportation as a server, the batch size will influence the transportation time. For example let us assume we have one person to move printed circuit boards from production to the testing department. When the batch size increases the load of the employee decreases and with it the waiting time in queue. For small batch sizes the system might even be unstable, see equation (1.66), and a second, costly employee would be necessary.

Thus move batches decrease the need for, and the load of, material handling, but they also increase the time in the system. From this dilemma an optimal batch size needs to be derived.
Chapter 3

Workload Controlled Manufacturing Systems

In this chapter we discuss Workload Controlled Manufacturing Systems (WCMS). In principle WCMS are of interest because by controlling the workload, or Work In Process, the time in the system can be reduced. An example is a Flexible Manufacturing System in which a fixed number of pallets transports some kind of material from one work station to another. This material is put on pallets at the load/unload station, and taken off again when it has completed all the required processing steps in the network. Clearly, the number of pallets limits the number of jobs in the system. If we assume that there is an infinite amount of raw material available so that pallets can always be loaded, we basically deal with a closed system with a fixed number of jobs. Alternatively, we could also imagine a manufacturing system where each job needs to be accompanied by a token and the amount of tokens is limited. Also Kanban controlled manufacturing systems are a form of WCMS since the number of Kanban cards is limited.

In this chapter we primarily deal with closed manufacturing systems. When all service times are exponential and, in case of multiple classes, class-independent, we can show again a product form solution, although in closed networks that does no longer imply statistical independence of the number of jobs at distinct workstations (Why not?). After discussing numerical solution approaches for product form networks, we turn to more general networks again and discuss approximations that are motivated by the exact algorithms for relatively simple systems.

3.1 Manufacturing systems modelled as Closed Queueing Networks

As we have mentioned in the introduction of this chapter closed manufacturing networks contain a constant number of jobs: no jobs arrive or depart. Consider again a flexible manufacturing system (FMS) consisting of a number of workstations that process jobs. For easy transportation and to facilitate operations, the jobs are fixed on pallets at a load/unload station. Assuming that always
unprocessed jobs are available this station, a finished job can be removed from the pallet and immediately replaced by a new job, so that the number of jobs is equal to the number of pallets present in the network. When all service times are exponential, we speak of Gordon-Newell networks, after the authors who were the first to show that for such networks a product form solution exists.

### 3.1.1 Gordon-Newell Networks

Gordon and Newell analyzed a specific type of Closed Queueing Networks (CQNs) which allowed a product-form solution of the stationary distribution of jobs in the network, see for instance [22]. These networks are also referred to as GN-networks. They are defined as follows.

**Definition 3.1** A GN-network is a single-class closed queueing network with \( M \geq 1 \) stations, \( N \geq 1 \) jobs. Station \( i \) has \( c_i \geq 1 \) servers each having exponentially distributed service times with mean value \( 1/\mu_i > 0 \). The service discipline employed is first-come first-served at all stations. The jobs have a Markovian routing, characterized by an irreducible routing matrix \( P \).

In the sequel we will tacitly assume that the load/unload station has index 0. Therefore the range of stations will be from \( 0 \leq i \leq M \).

As with open queueing networks the routing is described by the matrix \( P \). Here \( P_{0i} \) denotes the probability that a job is moved to the load/unload station upon completion at station \( i \). (Compare this to the meaning of \( P_{0i} \) in OQNs.)

In CQNs the visit ratios are derived similarly as in OQNs. Now the visit ratios \( V_i, i = 1, \ldots, M \) satisfy the homogeneous equation

\[
V_i = \sum_{j=1}^{M} V_j P_{ji}
\]

. Note that this equation only determines these ratios \( V_i \) up to a multiplicative constant. To fix this, we set the ratio of the load/unload station equal to 1, i.e., \( V_0 = 1 \). This makes sense as each pallet visits the load/unload station exactly once during a manufacturing cycle. Now we can define the throughput of the network as being equal to the load/unload station’s throughput. We explicitly assume that jobs do not need processing at the load/unload station as part of the processing chain, hence re-fitting or reorientation are not allowed.

**Ex. 3.1.1** Why is this last assumption necessary to allow setting the visit ratio of the load/unload station equal to 1?

**Ex. 3.1.2** What is \( P \) for a ring network with 4 stations in which all jobs travel in one direction?

**Ex. 3.1.3** What is \( P \) for the network of figure 3.1? Compute the visit ratios.

Let, as before \( n_i \) denote the number of jobs at station \( i, i = 0, \ldots, M \). The behavior of a GN-network can be described by a Markov process with states \( \vec{n} = (n_0, \ldots, n_M) \) and state space \( S_{GN} = \{ \vec{n} | \sum_{i=0}^{M} n_i = N \} \). The state space defines all the possible states that may occur in the network. Gordon and Newell...
3.1. MANUFACTURING SYSTEMS MODELLED AS CLOSED QUEUEING NETWORKS

Figure 3.1: This is an example of a closed network. The numbers indicate the fraction of jobs that, after being served, are routed to the same or another station. The stations are referenced by a letter to avoid confusion with the routing fractions. Station $f$ denotes the load/unload station.

found a product-form solution for GN-networks as is formulated in Theorem 3.2. Again this solution was found by inserting an educated guess in the balance equations.

**Theorem 3.2** *(Gordon and Newell [22])* Let $C$ be a GN-network with $M \geq 1$ stations and $N \geq 1$ jobs and mean service time at station $i$ equal to $1/\mu_i \geq 0$, $i = 0, \ldots, M$. Then the Markov process with states $n = (n_0, \ldots, n_M)$, with $n_i$ denoting the number of jobs at station $i$, and state space $S_{GN} = \{\vec{n} | \sum_{i=0}^{M} n_i = N\}$ has a unique stationary distribution:

$$
\pi(n_0, \ldots, n_M) = \frac{1}{G(M, N)} \prod_{i=0}^{M} f_i(n_i),
$$

where the normalization constant $G(M, N)$ is,

$$
G(M, N) = \sum_{\vec{n} \in S_{GN}} \prod_{i=0}^{M} f_i(n_i),
$$

and for each $i$, the function $f_i(n_i)$ is defined as

$$
f_i(n_i) = \frac{1}{\prod_{k=1}^{n_i} \min(k, c_i)} \left( \frac{V_i}{\mu_i} \right)^{n_i}, i = 0, \ldots, M.
$$

Note that this solution only depends on the routing matrix through the visit ratios. Thus two networks with different routing matrices but the same visit ratios will have the same stationary distribution of jobs in the network.

**Ex. 3.1.4** Find example networks that show the above phenomenon: two networks with different routing matrices but the same visit ratios will have the same stationary distribution of jobs in the network.

Equation (3.2) cannot be factorized into a product of single station normalization constants, as is the case in open Jackson networks. In the latter case a
station may contain any number of jobs. In a CQN this can certainly not be the case, as the number of jobs at one station depends on the number of jobs in the rest of network. Stated differently, the state space of one station can never be properly described in complete isolation of the rest of the network. The reader should reconsider the consequences of the assumptions underlying eq. (2.4) in view of the above.

In eq. (3.3) we tacitly assumed that the load/unload station with index 0 has a finite service rate. Often this is indeed the case, for instance when station 0 corresponds to a real machine. This is, however, not always true. When there is not a natural candidate machine or service station that can correspond to a load/unload station, it is customary to add such a, virtual, station at an appropriate location in the network. The throughput of this station can then be used to define the throughput of the entire network. Of course, a job’s service time at this virtual station should be 0; otherwise the virtual station would have a physically noticeable effect on the model of the network, which cannot be allowed. The probability ratio for a virtual station cannot be defined as in eq. (3.3), but should change to $f_0(n_0) = 0$, when $n_0 > 0$, and 1 otherwise. The reason is that, the service rate being infinite, there can never be a job in the station. In the sequel we assume that the load/unload station has finite service rate.

It is computationally very demanding to determine the normalization constant $G(M, N)$ directly from equation (3.2). An easier way to determine it is by the convolution algorithm, which is presented in Section 3.1.2. It is also possible to determine equations for the expected time in the system and number of jobs at each station by means of the so called Mean Value Analysis (MVA) algorithm, described in Section 3.1.3. However, the MVA algorithm is not valid for multiserver FCFS stations. In Section 3.1.4 we present the Marginal Distribution Analysis (MDA) algorithm by which it is possible to obtain the performance measures with multiple server stations. Finally in Section 3.1.5 we present a multi-class MDA algorithm.

Ex. 3.1.5 Compute $G(M, N)$ when $M = 2, N = 2$ where station 1 has 2 servers. (You should find 6 different states.)

Ex. 3.1.6 Show that the number of possible states, i.e., the cardinality of the set $S_{GN}$, is

$$\#S_{GN} = \binom{M + N}{N}.$$  

Note that we have $M + 1$ stations, that is, including the load/unload station with index 0. What is $\#S_{GN}$ when the load/unload station has infinite service rate?

### 3.1.2 The Convolution Algorithm

In this and the next section we present three algorithms to compute performance measures for Gordon-Newell networks: the Convolution algorithm, the Mean Value Analysis algorithm, and the Marginal Distribution algorithm. The main advantage of the first is that it enables us to obtain an expression for the probability of finding the network in a certain state, i.e., $\pi(n_0, \ldots, n_M)$, within a reasonable time. In other words, by means of the convolution algorithm we
3.1. MANUFACTURING SYSTEMS MODELLED AS CLOSED QUEUEING NETWORKS

can efficiently compute the normalization constant $G$, instead of having to sum the unscaled numbers $f_i(n_i)$ over the entire state space. The disadvantage of the Convolution algorithm as compared to the MVA and MDA algorithms is its greater sensitivity to rounding errors of floating point numbers. Therefore when it is only necessary to compute average performance indicators, such as $E_W$ or $E_L$, the MVA algorithm or, in the case of multiple servers per station the MDA algorithm, is more suitable.

**Performance measures expressed in terms of normalization constants**

Before we describe how to obtain the normalization constant $G$ by means of the convolution algorithm, we present simple expressions for some performance measures in terms of the normalization constant: the utilization $U_i$, the throughput $TH_i$ for station $i$, and, if each station has only one server, the expected number of jobs $E L_i$ and waiting time $E W_i$.

We make here a short note to interpret the throughput of a closed network in a slightly different way. We tag a job and have it travel around the network, that is, let it start at the load/unload station and wait until it enters this station again. It takes the tagged job on the average $\sum_{i=0}^{M} V_i E W_i$ units of time to make this trip through the network. By the time the tagged job has completed its journey, on average $N$ jobs, including the tagged jobs, will have passed the load/unload station once. Hence the load/unload station has processed $N$ jobs in $\sum_{i=0}^{M} V_i E W_i$ units of time, so that its throughput should be:

$$TH_0 = \frac{N}{\sum_{i=0}^{M} V_i E W_i}.$$  \hspace{1cm} (3.4)

In fact, this is Little’s law for a closed queueing network. Analogous expressions hold for the other stations, that is $TH_i = V_i TH_0$. Since the network is assumed to be in equilibrium, the rate at which jobs enter a station should be equal to the rate at which they depart. Hence these two rates are equal to the throughput of the station.

The expression for $\pi(\bar{n})$, where $\bar{n} = (n_0, \ldots, n_M)$, provides the key to finding the performance measures. Therefore we derive this expression now. Let us define for a network with $M + 1$ stations and $N$ jobs a shorthand for the state space of the network

$$S(M, N) = \left\{ \bar{n} \left| \sum_{j=0}^{M} n_j = N \right. \right\},$$

and

$$S(M \setminus \{i\}, N) = \left\{ \bar{n} \left| \sum_{j=0}^{M} n_j = N, n_i = 0 \right. \right\}$$

to denote the network with station $i$ removed. Analogously we define the normalization constants $G(M, N)$ and $G(M \setminus \{i\}, N)$. Note that the routing matrix will have to change to take the removal of station $i$ into account. We do this by defining a new routing matrix $P''$ as $P''_{j,k} = P_{ji}P_{ik} + P_{jk}$. 
Ex. 3.1.7 We could as well have set the service rate(s) of the server(s) of station \( i \) at \( \infty \). Why does this give, mathematically speaking, the the same result as the redefinition of \( P \)?

We know by the product form of the solution of Gordon-Newell networks that

\[
\pi(n_i = k) = \sum_{\substack{\vec{n} \in S(M,N) \, \vec{n}_i = k}} \pi(\vec{n})
\]

\[
= \frac{1}{G(M,N)} \sum_{\substack{\vec{n} \in S(M,N) \, \vec{n}_i = k}} \prod_{j=0}^{M} f_j(n_j)
\]

\[
= \frac{1}{G(M,N)} \sum_{\substack{\vec{n} \in S(M,N) \, \vec{n}_i = k}} f_i(k) \prod_{j \neq i} f_j(n_j)
\]

\[
= \frac{f_i(k)}{G(M,N)} \sum_{\substack{\vec{n} \in S(M \setminus \{i\}, N-k) \, \vec{n}_i = k}} \prod_{j \neq i} f_j(n_j)
\]

\[
= f_i(k) \frac{G(M \setminus \{i\}, N-k)}{G(M,N)}. \tag{3.5}
\]

From this equation we get an immediate recursion for \( G(M,N) \), since we have \( 1 = \sum_{k=0}^{N} \pi(n_i = k) \) so that,

\[
G(M,N) = \sum_{k=0}^{N} f_i(k) G(M \setminus \{i\}, N-k). \tag{3.6}
\]

This holds true for any station \( i \).

Ex. 3.1.8 Derive, by means of the above ideas, for a station with a single server:

\[
\mathbb{P}(n_i \geq k) = f_i(k) \frac{G(M,N-k)}{G(M,N)}.
\]

Does this expression also hold for multiserver stations?

Ex. 3.1.9 Imagine a circular network, i.e., jobs travel counter clockwise, say, from station \( n \) to \( n+1 \), and from \( M \) to 0. Each station has the same number of servers. All servers have equal service rate. What is your guess for \( \pi(\vec{n}) \) when there is one job in the network? Prove your guess with the equations above.

Having found the probability distribution in terms of normalization constants, we have relatively simple expressions for the utilization \( U_i \) and throughput \( TH_i \) for each station. \( U_i \) is easy:

\[
U_i = \mathbb{P}(n_i \geq 1) = 1 - \mathbb{P}(n_i = 0) = 1 - \frac{G(M \setminus \{i\}, N)}{G(M,N)}. \tag{3.7}
\]
3.1. MANUFACTURING SYSTEMS MODELLED AS CLOSED QUEUEING NETWORKS

The throughput requires slightly more work. We have motivated above that $TH_i$ equals the departure rate of jobs at station $i$, thus:

$$TH_i = \mu_i \sum_{k=1}^{N} \min(k, c_i) \mathbb{P}(n_i = k)$$

$$= \frac{\mu_i}{G(M, N)} \sum_{k=1}^{N} \min(k, c_i) f_i(k) G(M \{i\}, N - k)$$

Now recalling expression (3.3) we have that

$$\mu_i \min(k, c_i) f_i(k) = V_i f_i(k - 1), \quad k > 0.$$  

Therefore

$$TH_i = \frac{V_i}{G(M, N)} \sum_{k=1}^{N} f_i(k - 1) G(M \{i\}, N - k)$$

$$= \frac{V_i}{G(M, N)} \sum_{k=0}^{N-1} f_i(k) G(M \{i\}, N - k - 1)$$

$$= \frac{V_i G(M, N - 1)}{G(M, N)},$$  

(3.8)

by (3.6). Since the visit ratio of the load/unload station is 1, the system’s throughput is

$$TH_0 = \frac{G(M, N - 1)}{G(M, N)}.$$  

(3.9)

With Little’s formula we can compute the average time it takes to complete one tour through the network, which is

$$\mathbb{E}W = \frac{N}{TH_0} = N \frac{G(M, N)}{G(M, N - 1)}.$$  

(3.10)

The last performance measure to be derived is $EL_i$, whose definition can be restated:

$$\mathbb{E}L_i = \sum_{k=1}^{N} k \pi(n_i = k)$$

$$= \sum_{k=1}^{N} \mathbb{P}(n_i \geq k).$$

We therefore have for single server stations only

$$\mathbb{E}L_i = \frac{1}{G(M, N)} \sum_{k=1}^{N} f_i(k) G(M, N - k).$$  

(3.11)
CHAPTER 3. WORKLOAD CONTROLLED MANUFACTURING SYSTEMS

Obtaining the normalization constant

Having shown the use of the normalization constant in finding the performance metric, we will now derive an algorithm to compute it.

Equation 3.6 becomes for \( i = M \)

\[
G(M, N) = \sum_{k=0}^{N} f_M(k) G(M \setminus \{M\}, N - k),
\]

and is our fundamental recursion to find \( G(M, N) \). It is sometimes called Buzen’s formula. As always is the case when dealing with recursions, we need to define some initial conditions to drive it. These we will obtain now.

The first observation is that we do not need to consider negative values for \( M \) and \( N \). The second is that when there are no jobs in the network, there is only one configuration possible, which is trivial. Hence

\[
G(M, 0) = 1, \tag{3.12}
\]

for any \( M \). Finally, when there is only one station—which will, a fortiori, be the load/unload station—all jobs should reside at this station. Again there is only one possible distribution of the jobs over the stations, which is trivially,

\[
\pi(n_0 = N) = 1
\]

so that we should have for all \( N \), since the load/unload station is a single server station with \( \mu_0 < \infty \),

\[
G(0, N) = f_0(N) = \left( \frac{V_0}{\mu_0} \right)^N = \left( \frac{1}{\mu_0} \right)^N \tag{3.13}
\]

This is all there is; we have the starting values for Buzen’s formula.

Ex. 3.1.10 Compute \( G(2, 2) \) for single server stations.

We summarize the above method in the following algorithm.

Algorithm 1 Convolution Algorithm

Given \( M \) stations and 1 load/unload station with index 0, and \( N \) jobs.

1. (Initialization) Set
   \( (a) \) \( G(m, 0) = 1 \) for all \( 0 \leq m \leq M \).
   \( (b) \) \( G(0, n) = (1/\mu_0)^n \) for all \( 1 \leq n \leq N \).
   \( (c) \) \( n = 0 \)

2. \( n := n + 1, m := 0 \)

3. \( m := m + 1 \)

4. Compute

\[
G(m, n) = \sum_{k=0}^{n} f_m(k) G(m - 1, n - k)
\]
3.1. MANUFACTURING SYSTEMS MODELLLED AS CLOSED QUEUEING NETWORKS

5. If \( m < M \) then go to step 3, else go to step 6.
6. If \( n = N \) then stop; else go to step 2.

Ex. 3.1.11 Implement this algorithm in a computer program. (This is pretty straightforward and gives considerable insight in the method, and the results it provides.) Run it for some small \( M \) and \( N \), less than 30, say. What happens when \( M, N > 1000 \)?

Example 3.3 When there is only one machine per station we can simplify Buzen’s formula considerably, since

\[
G(M, N) = \sum_{k=0}^{N} f_M(k) G(M - 1, N - k)
\]

\[
= f_M(0) G(M - 1, N) + \sum_{k=1}^{N} f_M(k) G(M - 1, N - k).
\]

Since \( c_i = 1 \) by assumption, \( f_i(k) = (V_i/\mu_i)^k \). Taking \( i = M \) and shifting \( k \) by 1, the above becomes:

\[
G(M, N) = G(M - 1, N) + \frac{V_M}{\mu_M} \sum_{k=0}^{N-1} f_M(k) G(M - 1, N - k - 1).
\]

The second term can now be recognized as \( G(M, N - 1) \), by which we find the final result for single machine stations,

\[
G(M, N) = G(M - 1, N) + \frac{V_M}{\mu_M} G(M, N - 1).
\]

Ex. 3.1.12 Why can this result not be generalized to multiserver stations?

3.1.3 Mean Value Analysis algorithm for single server station networks

The Mean Value Analysis (MVA) algorithm is a recursive procedure to compute the performance measures of a closed network. Be aware that this procedure may only be applied to single server stations (see the next sections for the analysis of more general cases). The recursion starts by finding the measures with only one job in the system. After each iteration a job is added and the measures are computed again. This process continues until all \( N \) jobs have been added to the network.

First we will introduce some shorthands. With \( n \) jobs in the network:

- \( L_j(n) \) = the number of jobs at station \( j \)
- \( W_j(n) \) = the time in the station \( j \)
- \( TH_j(n) \) = the throughput of station \( j \)
• $S_j =$ the service time of station $j$.

Note that $L_j(n)$, $W_j(n)$ and $S_j$ are all random variables.

Each iteration of the MVA algorithm requires four consecutive steps. Specifically, the steps involve the computation of the expectation of the:

1. time in each station
2. throughput of the network
3. throughput per station
4. number of jobs at each station.

The expected throughput of the network and the expected number of jobs at each station follow from Little’s law. The equation for the expected time in each station is based on the Arrival Theorem. Roughly spoken, this theorem states that an arriving job at any workstation observes the whole system as being a system in equilibrium (steady-state) with one job less. Although the theorem holds for any product-form network, we prove it only for the case of single class networks with single server workstations.

**Theorem 3.4 (Arrival Theorem)** When a job travels from station $i$ to station $j$ in a closed queueing network with $N$ jobs and exponentially distributed service times, the stationary distribution of the number of jobs that the traveling job observes is equal to the stationary distribution of the same closed queueing network with $N - 1$ jobs (i.e. with one job less).

**Proof.** (From [66], only for networks with single server workstations) Consider a job that has just left station $i$ and is headed to station $j$. Let us determine the probability of the system as seen by this job. In particular, let us determine the probability that this job observes, at that moment, $n_l$ jobs at server $l$, $0 \leq l \leq M$, $\sum_{l=0}^{M} n_l = N - 1$. This is done as follows:

$$P\{\text{job observes } n_l \text{ jobs at server } l| \text{job goes from } i \text{ to } j\} = \frac{P\{\text{job goes from } i \text{ to } j\}}{P\{\text{state is } (n_0, \ldots, n_i + 1, \ldots, n_j, \ldots, n_M), \text{ job goes from } i \text{ to } j\}}$$

$$= \frac{\pi_{\langle N \rangle}\{(n_0, \ldots, n_i + 1, \ldots, n_j, \ldots, n_M)\} \mu_i P_{ij}}{\sum_{\mathbf{n} \in S(M, N-1)} \pi_{\langle N \rangle}\{(n_0, \ldots, n_i + 1, \ldots, n_M)\} \mu_i P_{ij}}$$

$$= \frac{(V_i/\mu_i) \prod_{j=0}^{M} (V_j/\mu_j)^{n_j}}{K}, \text{ by eq.(3.1)}$$

$$= C \prod_{j=0}^{M} (V_j/\mu_j)^{n_j}$$

where the constants $C$ and $K$ do not depend on $\mathbf{n}$, and the subscript $\langle N \rangle$ denotes that we deal with a job population of size $N$. But as the above is a probability density on the set of vectors $\mathbf{n}$, $\sum_{j=0}^{M} n_j = N - 1$, it follows from eq.(3.1) that it must be equal to $\pi_{\langle N-1 \rangle}\{(n_0, \ldots, n_M)\}$. This establishes our claim. ■
First we will derive an equation for the expected time in station $j$. We have seen earlier that $\mathbb{E}W_j$ is the sum of the expected waiting time in queue plus the expected service time. When there are $n$ jobs present, we therefore have

$$\mathbb{E}W_j(n) = \mathbb{E}W_{Qj}(n) + \mathbb{E}S_j.$$  
(3.14)

The expected service time is equal to $1/\mu_j$, independent of the arrival time of the job. (Remember that the exponential distribution is memoryless).

The expected waiting time in the queue is the time the job will spend in queue until it can be serviced. According to the arrival theorem the job arriving at station $j$ sees the station in its stationary state as if there were $n-1$ jobs present in the network. Thus the job expects $\mathbb{E}L_j(n-1)$ in $j$. From this we find for the expected waiting time for an arriving job (recall we deal with single server stations)

$$\mathbb{E}W_{Qj}(n) = \mathbb{E}L_j(n-1) \mathbb{E}S_j,$$  
(3.15)

The expected time in station $j$ in a network with $n$ jobs becomes now:

$$\mathbb{E}W_j(n) = \mathbb{E}L_j(n-1) \mathbb{E}S_j + \mathbb{E}S_j$$

$$= (\mathbb{E}L_j(n-1) + 1) \mathbb{E}S_j.$$  
(3.16)

This equation is only valid for single server stations (Why?), so that the applicability of the MVA algorithm is restricted to this case only.

The second step in the recursion determines the throughput of the network. Note that the throughput of the network is equal to the throughput of the load/unload station because we have set the visit ratio of the load/unload station equal to 1. We find the network's throughput by applying Little's law at system level,

$$TH_0(n) = \frac{n}{\sum_{i=0}^{M} V_i \mathbb{E}W_i(n)}.$$  
(3.17)

The throughput of station $j$ should therefore be

$$TH_j(n) = V_j TH_0(n), \quad j = 0, \ldots, M,$$  
(3.18)

yielding the third algorithmic step.

The fourth and final step involves the equation for the expected number of jobs at station $j$ with $n$ jobs in the network. Applying Little's law to a single station:

$$\mathbb{E}L_j(n) = TH_j(n) \mathbb{E}W_j(n), \quad j = 0, \ldots, M.$$  
(3.19)

Based on equations (3.16) through (3.19) we can design a recursive algorithm that computes the performance measures for single server closed queueing networks. The boundary condition of the recursion is found by noting that when there is only one job in the network this job never has to wait in queue. The MVA algorithm for single server GN-networks is formulated in Algorithm 2.

**Algorithm 2 Mean Value Analysis**

1. (Initialization) Set $V_0 = 1$. Determine the other visit ratio's $V_j, j = 1, \ldots, M$, from the set of linear equations given by $V_j = \sum_{i=0}^{M} V_i P_{ij}$. Set $n = 0$ and $\mathbb{E}L_j(n) = 0, j = 0, \ldots, M$. 


2. \( n := n + 1 \)

3. Compute \( \mathbb{E}W_j (n) \) for \( j = 0, \ldots, M \) from:
   \[
   \mathbb{E}W_j (n) = (\mathbb{E}L_j (n-1) + 1) \mathbb{E}S_j .
   \]

4. Compute \( TH_0 (n) \) from:
   \[
   TH_0 (n) = \frac{n}{\sum_{j=0}^{M} V_j \mathbb{E}W_j (n)} .
   \]
   and compute \( TH_j (n) \) for \( j = 1, \ldots, M \) from:
   \[
   TH_j (n) = V_j TH_0 (n) .
   \]

5. Compute \( \mathbb{E}L_j (n) \) for \( j = 0, \ldots, M \) from:
   \[
   \mathbb{E}L_j (n) = TH_j (n) \mathbb{E}W_j (n) .
   \]

6. If \( n = N \) then stop; else go to step 2.

---

**Example 3.5**

Consider a Gordon-Newell network with \( M = 5 \) stations in tandem, that is, station \( n \) outputs all its jobs to station \( n + 1 \), and station 4 feeds into station 0. Suppose the service rates are \( \mu_0 = 2, \mu_1 = 2.5, \mu_2 = \mu_3 = \mu_4 = 3 \) (unit in jobs/day, say). Figure 3.2 shows \( \mathbb{E}L_1, \mathbb{E}L_2 \), and the network’s throughput \( TH_0 \) as computed with the MVA algorithm.

**Ex. 3.1.13** Why is \( \mathbb{E}L_1 \approx 4 \) when \( N > 30 \), say?

**Ex. 3.1.14** How many jobs does station 0 approximately contain as a function of \( N \) when \( N > 30 \)?

We state two rules of thumb for closed networks in which the servers have equal visit ratios but different service rates:

- The network’s throughput is dominated by the service rate of the slowest server;
- Most of the jobs will be queued in the buffer at the slowest server.

From this we claim that when it is necessary to optimize the system’s throughput or reduce the probability of queue overflow, one only has to spend time and money to:

- Locate the slowest server;
- Once located, improve the service rate of just this server.

Trying to increase the throughput by working on any server that has higher, even just a little, service rate is non-optimal.
3.1. MANUFACTURING SYSTEMS MODELLED AS CLOSED QUEUEING NETWORKS

![Network throughput and expected number of jobs in station 1 and 2.](image)

Figure 3.2: The network’s throughput and expected number of jobs in station 1 and 2. Clearly, $TH_0$ and $E_{L_2}$ approach 2 as $N$ increases.

**Ex. 3.1.15** Refine this claim by making some of the assumptions explicit. For instance, does the claim remain valid, if you change:

- The routing matrix of connection between the stations?
- The number of jobs in the network, e.g $N \ll M$, $N \approx M$, or $N \gg M$?
- The service rate at all servers such that they have approximately the same service rate?

The interested reader should consult Kleinrock [45, Volume 2] for a much more detailed analysis of this example.

3.1.4 Marginal Distribution Analysis Algorithm

The MVA algorithm as presented in Section 3.1.3 is only valid for GN-networks with single server stations. Although equations (3.17, 3.18) and (3.19) still hold for standard GN-networks, equation (3.16) for the expected time in a station is no longer true for multiserver stations. The reason is that for multiserver stations the expected waiting time is not simply equal to the product of the expected number of jobs at the station and the expected service time. Still, the arrival theorem is valid which we therefore exploit to derive a more detailed recursive analysis. This recursive relation enables us to compute the marginal probabilities of the number of jobs at each station; hence the name Marginal Distribution Analysis (MDA) algorithm.

Before continuing we introduce, again, some notation:

- $n = \text{the number of jobs in the system}$
• \( p_j(k|n) \) = the marginal (stationary) probability of having \( k \) jobs at station \( j \) with \( n \) jobs in the network

• \( D_{rem,j}(n) \) the time between the arrival of a job at station \( j \) and the first departure at station \( j \)

• \( TCQ_j(n) \) = the time to clear the queue at station \( j \)

• \( Q_j(n) \) = the probability that all servers are busy at station \( j \)

Note that \( p_j(k|n) \) does not say anything about the number of jobs at the other stations besides station \( j \).

Now we will derive the equation for the expected waiting time at station \( j \). When a job arrives at station \( j \) when less than \( c_j \) jobs are present it does not have to wait in queue. On the other hand, when all servers are occupied, it has to wait for the first departure after all the jobs in front of it have been taken into service. We can split this time into the expected time until the next job leaves, \( \mathbb{E}D_{rem,j}(n) \), and the expected time to clear the queue, \( \mathbb{E}(TCQ_j(n)) \).

The expected time until the next job leaves follows simply by conditioning on whether all servers are busy:

\[
\mathbb{E}D_{rem,j}(n) = \mathbb{E}(D_{rem,j}(n)| \text{At least 1 server free}) P(\text{At least 1 server free}) \\
+ \mathbb{E}(D_{rem,j}(n)| \text{All servers busy}) P(\text{All servers busy}) \\
= \mathbb{E}S_{rem,j}(n) P(\text{All servers busy}), \tag{3.20}
\]

since \( \mathbb{E}(D_{rem,j}(n)| \text{At least 1 server free}) = 0 \). Furthermore we write \( \mathbb{E}S_{rem,j}(n) \) for \( \mathbb{E}(D_{rem,j}(n)| \text{All servers busy}) \). Let the service times be exponentially distributed with parameter \( \mu_j \) and let there be \( c_j \) servers at station \( j \). Then, when all servers are busy, the time between two departures is \( 1/(c_j\mu_j) \). Because of the memoryless property of the exponential distribution the time until the first departure is \( 1/(c_j\mu_j) \) as well. The probability that all servers are busy at station \( j \) when a job arrives, is given by:

\[
Q_j(n-1) = \sum_{k=c_j}^{n-1} p_j(k|n-1) \tag{3.21}
\]

Note that we applied the arrival theorem. Now we find for the expected time until the first departure:

\[
\mathbb{E}D_{rem,j}(n) = \mathbb{E}(\mathbb{E}(\text{Time to first departure}| \text{all servers busy})) \\
= Q_j(n-1) \mathbb{E}S_{rem,j}(n) \\
= \sum_{k=c_j}^{n-1} p_j(k|n-1) \frac{1}{c_j\mu_j}, \tag{3.22}
\]

The expected time to clear the queue, given that the arriving job sees \( k \geq c_j \) jobs at station \( j \), can be found in the same way to be

\[
\frac{k-c_j}{c_j\mu_j}
\]
Then the expected time to clear the queue is:

\[ \mathbb{E} TCQ_j (n) = \sum_{k=c_j}^{n-1} \frac{k-c_j}{c_j \mu_j} p_j (k\mid n-1). \] (3.23)

Also here the arrival theorem is applied.

The expected waiting time at station \( j \) is therefore

\[ \mathbb{E} W_{Q,j} (n) = \mathbb{E} S_{rem,j} (n) + \mathbb{E} TCQ_j (n) = \sum_{k=c_j}^{n-1} \frac{k-c_j+1}{c_j \mu_j} p_j (k\mid n-1). \] (3.24)

By substituting \( c_j = 1 \) it is easy to see that the equation reduces to equation (3.15) used for single server stations with the MVA algorithm.

The expected time of a job in station \( j \) becomes

\[ \mathbb{E} W_j (n) = \sum_{k=c_j}^{n-1} \frac{k-c_j+1}{c_j \mu_j} p_j (k\mid n-1) + \frac{1}{\mu_j}. \] (3.25)

To complete the MDA algorithm we need equations for the marginal probabilities \( p_j (k\mid n) \). We can find these by noting that the rate with which the station \( j \) changes state from \( k \) to \( k-1 \) should equal the rate from \( k-1 \) to \( k \). This observation is based on the principle that when the system is in stationary state, the balance equation check (compare this to the Markov chain of the M/M/1 queue). Let us first consider the rate from \( k-1 \) to \( k \). It is obvious that the throughput, and thus the arrival rate, of the jobs at station \( j \) is equal to \( TH_j (n) \). By the arrival theorem an arriving job sees the system as having \( n-1 \) jobs. Thus the rate from \( k-1 \) to \( k \) should be:

\[ TH_j (n) \ p_j (k-1\mid n-1). \]

On the other hand, the rate from \( k \) to \( k-1 \) is of course,

\[ \mu_j \ \min (c_j, k) \ p_j (k\mid n) \]

Thus with the two rates being equal we arrive at

\[ \mu_j \ \min (c_j, k) \ p_j (k\mid n) = TH_j (n) \ p_j (k-1\mid n-1), \quad k = 1, \ldots, N. \] (3.26)

The probabilities \( p_j (0\mid n) \) follow from the fact that the sum of the probabilities is 1:

\[ p_j (0\mid n) = 1 - \sum_{k=1}^{n} p_j (k\mid n). \] (3.27)

These probabilities satisfy, again, a recursion in \( n \) and \( k \).

With the above recursions for the performance measures we now formulate the MDA algorithm. Note that the algorithm reduces to the MVA algorithm when \( c_j = 1 \).

**Algorithm 3** Marginal Distribution Analysis
1. (Initialization) Set $V_0 = 1$. Determine the other visit ratios $V_j, j = 1, \ldots, M$. Set $n = 0$ and $p_j (0|0) = 1, j = 0, \ldots, M$.

2. $n := n + 1$

3. Compute $E W_j (n)$ for $j = 0, \ldots, M$ from:

   $$E W_j (n) = \sum_{k=c_j}^{n-1} \frac{k - c_j + 1}{c_j \mu_j} p_j (k|n - 1) + \frac{1}{\mu_j}.$$  

4. Compute $T H_0 (n)$ from:

   $$T H_0 (n) = \frac{n}{\sum_{j=0}^{M} V_j E W_j (n)}.$$  

and compute $T H_j (n)$ for $j = 1, \ldots, M$ from:

   $$T H_j (n) = V_j T H_0 (n).$$

5. Compute the marginal probabilities $p_j (k|n)$ for $k = 1, \ldots, n$ and $j = 0, \ldots, M$ from:

   $$\mu_j \min (c_j, k) p_j (k|n) = T H_j (n) p_j (k-1|n-1),$$  

and compute $p_j (0|n)$ for $j = 0, \ldots, M$ from:

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

6. If $n = N$ then stop; else go to step 2.

### 3.1.5 Multi-Class MDA

In this section we will introduce multiple class CQNs. In 1975 Baskett, Chandy, Muntz and Palacios [7] extended the work of Gordon and Newell to multi-class CQNs, which was only valid for FCFS servers, to include also the Processor Sharing, Ample Server and Last-Come First-Served Preemptive Resume scheduling. Their result, again a product form solution, is commonly referred to as the BCMP theorem. The networks satisfying the involved conditions are known as BCMP-networks and are defined as follows.

**Definition 3.6** A BCMP-network is a multi-class closed queueing network with $M \geq 1$ stations, $R \geq 1$ job classes, $N^{(r)} \geq 1$ class-$r$ jobs. Each station has any of the following four service disciplines:

- **FCFS** First Come First Served,
- **LCFS** Last Come First Served-Preemptive Resume,
- **PS** Processor Sharing,
3.1. MANUFACTURING SYSTEMS MODELLED AS CLOSED QUEUEING NETWORKS

AS Ample Server.

The service times of class-\( r \) jobs at station \( i \) have mean value \( 1/\mu_{i}^{(r)} \geq 0 \). For LCFS-PR, PS, and AS, the service times can be general and class-dependent. For FCFS, the service times should be exponential and class-independent. The jobs have a Markovian routing, characterized by the irreducible routing matrix \( P^{(r)} \) for class \( r \).

**Ex. 3.1.16** Give an intuitive argument why LCFS-PR, PS, and AS, can allow for more general service disciplines than FCFS.

From Definition 3.6 we see that GN-networks are a special case of BCMP-networks.

In this section we are only concerned with the FCFS service discipline where all job classes have mean service time \( 1/\mu_{i} \) at station \( i \).

The BCMP theorem is only concerned with the equilibrium distribution of the stochastic process obtained by aggregating the states of the Markov process with the same number of jobs per class. This means that only the number of jobs per class in queue is relevant, but not the exact sequence of these jobs in queue. For the state description of this stochastic process, we introduce a vector notation. Its state space is given by

\[
\mathcal{S}_{BCMP} = \left\{ (\bar{n}_1, \ldots, \bar{n}_M) \mid \sum_{i=1}^{M} n_i^{(r)} = N^{(r)}, r = 1, \ldots, R \right\}
\]

in which \( \bar{n}_i = (n_i^{(1)}, \ldots, n_i^{(R)}) \) and \( n_i^{(r)} \) denotes the number of class-\( r \) jobs at station \( i \).

The population of the network is given by \( \bar{N} = (N^{(1)}, \ldots, N^{(R)}) \). Furthermore, we need to introduce the visit ratio’s \( V_{j}^{(r)} \) of station \( j \) of job type \( r \). These ratio’s are obtained from the equations

\[
\bar{V}^{(r)} = \tilde{V}^{(r)} P^{(r)},
\]

where \( \tilde{V}^{(r)} = (V_1^{(r)}, \ldots, V_M^{(r)}) \), and by setting \( V_0^{(r)} = 1 \). In general we will add a superscript \( r \) to a performance measure to denote that it belongs to class \( r \) jobs. For instance, \( TH_0^{(r)}(\bar{n}) \) is the throughput of class \( r \) jobs through station 0 (which is thereby per definition the network’s throughput).

**Theorem 3.7** (Baskett, Chandy, Muntz and Palacios [7]) Let \( C \) be a BCMP-network with \( M \) stations, \( R \) job classes, \( N^{(r)} \) jobs in class \( r \) and with visit ratio’s \( V_i^{(r)} \). Then the detailed Markov process that describes the behavior of \( C \) has a unique stationary distribution and the aggregated stationary probabilities \( \pi(\cdot) \) for the aggregate states \( \bar{n} = (\bar{n}_1, \ldots, \bar{n}_M) \) are given by:

\[
\pi(\bar{n}_1, \ldots, \bar{n}_M) = \frac{1}{G(\bar{N})} \prod_{i=1}^{M} f_i(\bar{n}_i)
\]

where the function \( G(\bar{N}) \) is determined from the normalization:

\[
G(\bar{N}) = \sum_{\bar{n} \in \mathcal{S}_{BCMP}} \prod_{i=1}^{M} f_i(\bar{n}_i)
\]

and the function \( f_i(\bar{n}_i) \) is given by

\[
f_i(\bar{n}_i) = \begin{cases} 
\frac{n_i!}{\prod_{k=1}^{n_i} \min(k, c_i)} \prod_{r=1}^{R} \frac{1}{n_i^{(r)}!} \left( \frac{V_i^{(r)}}{\mu_i^{(r)}} \right)^{n_i^{(r)}} & i \text{ is FCFS, LCFS, PS.} \\
\prod_{r=1}^{R} \frac{1}{n_i^{(r)}!} \left( \frac{V_i^{(r)}}{\mu_i^{(r)}} \right)^{n_i} & i \text{ is AS.}
\end{cases}
\]
where \( n_i = \sum_{r=1}^{R} n_{i}^{(r)} \) denotes the total number of jobs at station \( i \).

From the stationary distribution \( \pi(\tilde{n}_1, \ldots, \tilde{n}_M) \), we can, in principle, obtain the relevant performance measures such as the mean number of jobs and the waiting time at a station by means of the normalization constant. However, as was the case with single class CQNs, there are too many states to be able to compute this within reasonable time. To find the measures we will make use of a generalized arrival theorem. Since there still is which is valid for BCMP-networks. This generalization was first presented by Lavenberg and Reiser [48], and is formulated as follows.

**Theorem 3.8 (The Generalized Arrival Theorem)** Let \( C(\tilde{N}) \) be a BCMP-network with population \( \tilde{N} \). Denote by \( p(\tilde{n}_1, \ldots, \tilde{n}_M) \) the equilibrium probability of \( C(\tilde{N}) \), and by \( p^{(r)}(\tilde{n}_1, \ldots, \tilde{n}_M) \) the equilibrium probability that, at a class-\( r \) arrival instant at an arbitrary station, the state of \( C(\tilde{N}) \) is \( (\tilde{n}_1, \ldots, \tilde{n}_M) \).

Then:

\[
p^{(r)}(\tilde{n}_1, \ldots, \tilde{n}_M) | \tilde{N} = p((\tilde{n}_1, \ldots, \tilde{n}_M) | \tilde{N} - \tilde{e}_r),
\]

where \( \tilde{e}_r \) is the \( R \)-dimensional unit vector with 1 at position \( r \).

The proof of this theorem can be found in for instance [38] and is an extension of the proof of the arrival theorem 3.4.

With the generalized arrival theorem we can generalize algorithm 3 to a multi-class MDA algorithm, see below.

**Algorithm 4 Multi-Class MDA Algorithm**

1. (Initialization) Set \( V_0^{(r)} = 1 \). Determine the other visit ratio’s \( V_j^{(r)} \), \( j = 1, \ldots, M \) and \( r = 1, \ldots, R \). Set \( n = 0 \) and \( p_j(0|\tilde{n}) = 1, j = 0, \ldots, M \).

2. \( n := n + 1 \)

3. For all states \( \tilde{n} \in \left\{ \tilde{n} \mid \sum_{r=1}^{R} n^{(r)} = n \text{ and } n^{(r)} \leq N^{(r)} \right\} \) execute steps 4 through 6.

4. Compute \( EW_j^{(r)}(n) \) for \( j = 0, \ldots, M \) and \( r = 1, \ldots, R \) (for which \( n^{(r)} > 0 \) and \( V_j^{(r)} > 0 \)) from:

\[
EW_j^{(r)}(\tilde{n}) = \sum_{k=c_j}^{n-1} \frac{k - c_j + 1}{c_j\mu_j} p_j(k|\tilde{n} - \tilde{e}_r) + \frac{1}{\mu_j}.
\]

5. Compute \( TH_0^{(r)}(\tilde{n}) \) for \( r = 1, \ldots, R \) if \( n^{(r)} > 0 \) from:

\[
TH_0^{(r)}(\tilde{n}) = \frac{n^{(r)}}{\sum_{j=0}^{M} V_j^{(r)} EW_j^{(r)}(\tilde{n})},
\]

and if \( n^{(r)} = 0 \) then \( TH_0^{(r)}(\tilde{n}) = 0 \). Compute \( TH_j^{(r)}(\tilde{n}) \) for \( j = 1, \ldots, M \) and \( r = 1, \ldots, R \) from:

\[
TH_j^{(r)}(\tilde{n}) = V_j^{(r)} TH_0^{(r)}(\tilde{n})).
\]
3.1. MANUFACTURING SYSTEMS MODELLED AS CLOSED QUEUEING NETWORKS

Compute $TH_j (\bar{n})$ for $j = 1, \ldots, M$ from

$$TH_j (\bar{n}) = \sum_{r=1}^{R} TH_j^{(r)} (\bar{n}).$$

6. Compute the marginal probabilities $p_j (k) | \bar{n}$ for $k = 1, \ldots, \bar{n}$ and $j = 0, \ldots, M$ if $TH_j (\bar{n}) > 0$ from:

$$\mu_j \min (c_j, k) \ p_j (k) | \bar{n} = \sum_{r=1}^{R} TH_j^{(r)} (n) \ p_j (k-1) | \bar{n} - \bar{c}_r,$$

and if $TH_j (\bar{n}) = 0$ then $p_j (k) | \bar{n} = 0$. Compute $p_j (0) | \bar{n}$ for $j = 0, \ldots, M$ from:

$$p_j (0) | \bar{n} = 1 - \sum_{k=1}^{\bar{n}} p_j (k) | \bar{n}.$$

7. If $n = N$ then stop; else go to step 2.

Ex. 3.1.17 Think about the cardinality of the state space defined in step 3 of the algorithm, and compare it to that of the state space of GN-networks, $S_{GN}$, see exercise 3.1.6. What do you conclude for the algorithmic complexity of the multi-class MDA algorithm?

3.1.6 Restricted Open Queueing Networks

In restricted open queueing networks, the number of jobs is not necessarily fixed but bounded from above, i.e., there is a maximum number of jobs that the network may contain. As an example imagine again a flexible manufacturing system. In the previous section we modeled this as a closed network. The critical assumption there was that always unprocessed jobs had to available at the load/unload station to enter the network for service. In general this assumption need not be satisfied. It can happen that pallets—representing the jobs of the closed queueing network—are waiting at the load/unload station for new jobs to arrive. Still we can assume that there is some maximum number of pallets available to handle jobs. Hence this type of network shares some characteristics with open networks, as the number of jobs in the network is variable. On the other hand, the number of jobs is restricted to the number of available pallets.

Interestingly, these restricted open networks can be modeled as closed queueing networks. To understand this, consider the situation in which a few pallets are idle, i.e., waiting at the load/unload station. If we assume that jobs out of the network arrive at the load/unload station according to a Poisson process with rate $\gamma_0$, then pallets leave this station at the same rate $\gamma_0$. Therefore we might as well define the service time of the load/unload station as being exponentially distributed with rate $\gamma_0$, as long as pallets have to queue at the load/unload station. On the other hand, when jobs arriving from outside are allowed to queue, and wait until a pallet arrives, a pallet can be loaded immediately. If we assume that the load/unload time is zero, the pallet will leave
station 0 without any delay. Clearly, the service rate at the load/unload station can no longer be exponential with rate $\gamma_0$.

To simplify the analysis we assume that jobs arriving at station 0 when no pallet is available are rejected so that no external job queue is build up. Now a pallet will at its arrival at the load/unload station never find a job to be loaded, and hence will have to wait an exponential time before it can leave (why?). As a result of this assumption, the entire analysis of closed queueing networks carries over to this case with station 0 having service rate $\gamma_0$.

If the loading and unloading at station 0 takes a non-negligible time, anyhow (even if always jobs are available), the service rate will not be so clear anymore if we let pallets queue at station 0 as well. The trick to deal with this case, is to insert a virtual station, just before the load/unload station, and imagine that pallets will queue there and wait for external jobs to arrive. The real load/unload station can now become a normal station, requiring a certain service time.

Ex. 3.1.18 Suppose jobs are queued instead of rejected. What can you say in qualitative terms about the network’s throughput?

3.2 General workload controlled manufacturing systems

We can model Workload Controlled Manufacturing Systems (WCMS) as closed queueing networks when we assume that there is always an unprocessed job available to enter the network when a finished job leaves. Another form of WCMS are the restricted open queueing networks. In Subsection 3.1.6 we have seen that we can model ROQNs with Poisson arrival streams as CQNs as well.

As mentioned earlier, in real-world manufacturing systems it is often the case that the service times are not exponentially distributed or class-independent when there are multiple classes of jobs. The performance measures of WCMS can then still be approximated by assuming that the product form solution, and with it the arrival theorem, is still valid in CQNs with service times that have a general distribution and are class-dependent.

First we will extend the single class Marginal Distribution Analysis algorithm to a single class Approximate Mean Value Analysis algorithm (AMVA), with which we can analyze single class WCMS. In Section 3.2.2 we will extend the single class AMVA together with the multi-class MDA algorithm to a multi-class AMVA algorithm.

3.2.1 Single Class WCMS

To extend Algorithm 3 to approximate performance measures for CQNs with general distributed service times the equation for the time at a station, eq. (3.25), and the equation for marginal probabilities, eq. (3.26), have to be adjusted.

In Section 3.1.4 we have seen that the time at a station can be split up in three stages: the delay until the first departure $D_{rem,j}$, the time to clear the queue $TCQ_j$ and the service time of the job itself. Since the service times no longer have an exponential distribution we can not use equations (3.22, 3.23) for the $ED_{rem,j}$ and $ETCQ_j$ anymore. Here we will suggest alternative equations.
3.2. GENERAL WORKLOAD CONTROLLED MANUFACTURING SYSTEMS

The delay until the first departure is given by the product of the probability that all servers are busy when a job arrives, \( Q_j (n-1) \), and time until the first departure, \( \mathbb{E}S_{\text{rem},j} \). The probability that all servers are busy is still given by eq. (3.21). Buitenhek [15] proposes to approximate the time between an arrival and the first departure given that all servers are busy by the corresponding expression for the M/G/c queue given by Tijms [80]:

\[
\mathbb{E}S_{\text{rem},j} = (1 - C_{sj}^2) \frac{\mathbb{E}S_j}{c_j + 1} + C_{sj}^2 \frac{\mathbb{E}S_j}{c_j}
\]  

Using the definition of the SCV we can rewrite this as

\[
\mathbb{E}S_{\text{rem},j} = \frac{c_j - 1}{c_j + 1} \frac{\mathbb{E}S_j}{c_j} + \frac{2}{c_j + 1} \frac{1}{c_j} \frac{\mathbb{E}S_j^2}{2\mathbb{E}S_j}
\]

The expected time to clear the queue is given by the product of the expected number of jobs in the queue when a job arrives, \( \mathbb{E}L_{Q,j}(n-1) \), and the mean service time of the jobs divided by the number of servers, \( \mathbb{E}S_j/c_j \). Hence,

\[
\mathbb{E}TCQ_j (n) = \mathbb{E}L_{Q,j}(n-1) \frac{\mathbb{E}S_j}{c_j}
\]

where

\[
\mathbb{E}L_{Q,j}(n-1) = \sum_{k=c_{j+1}}^{n-1} (k - c_j) p_j (k \mid n-1).
\]

Thus the expected time at any station \( j \) can is

\[
\mathbb{E}W_j(n) = Q_j (n-1) \mathbb{E}S_{\text{rem},j} + \mathbb{E}L_{Q,j}(n-1) \frac{\mathbb{E}S_j}{c_j} + \mathbb{E}S_j
\]

The marginal probabilities can be approximated by substituting \( \mathbb{E}S_j \) for \( 1/\mu_j \) in equation (3.26). Now we can formulate an algorithm to analyze a single class CQN with general distributed service times, see Algorithm 5.

**Algorithm 5 Single Class Approximate Mean Value Analysis**

1. (Initialization) Set \( V_0 = 1 \). Determine the other visit ratio’s \( V_j, j = 1, \ldots, M \). Set \( n = 0 \) and \( p_j (0 \mid 0) = 1, j = 0, \ldots, M \).
2. \( n := n + 1 \)
3. Compute \( \mathbb{E}W_j(n) \) for \( j = 0, \ldots, M \) from

\[
\mathbb{E}W_j(n) = Q_j (n-1) \mathbb{E}S_{\text{rem},j} + \mathbb{E}L_{Q,j}(n-1) \frac{\mathbb{E}S_j}{c_j} + \mathbb{E}S_j,
\]

where

\[
Q_j (n-1) = \sum_{k=c_j}^{n-1} p_j (k \mid n-1),
\]

\[
\mathbb{E}S_{\text{rem},j} = \frac{c_j - 1}{c_j + 1} \frac{\mathbb{E}S_j}{c_j} + \frac{2}{c_j + 1} \frac{1}{c_j} \frac{\mathbb{E}S_j^2}{2\mathbb{E}S_j},
\]

\[
\mathbb{E}L_{Q,j}(n-1) = \sum_{k=c_{j+1}}^{n-1} (k - c_j) p_j (k \mid n-1)
\]
4. Compute \( TH_0(n) \) from
\[
TH_0(n) = \sum_{i=0}^{M} \frac{n}{V_i \mathbb{E}[W_i(n)]},
\]
and compute \( TH_j(n) \) for \( j = 1, \ldots, M \)
\[
TH_j(n) = V_j \; TH_0(n).
\]

5. Compute \( p_j(k|n) \) for \( k = 1, \ldots, n \) and \( j = 0, \ldots, M \) from
\[
\min(c_j,k) \frac{\mathbb{E}[S_j(n)]}{p_j(k|n)} = TH_j(n) \; p_j(k-1|n-1),
\]
\[
p_j(0|n) = 1 - \sum_{k=1}^{n} p_j(k|n).
\]

6. If \( n = N \) then stop; else go to step 2.

### 3.2.2 Multi-class WCMS

In this section we extend the single class AMVA algorithm together with the multi-class MDA algorithm to a multi-class AMVA algorithm with reduced complexity, where jobs may have general distributed, class-dependent service times. We will reduce the complexity to make the algorithm computationally more attractive. First we will show how class-dependent service times can be incorporated by using aggregate service times.

**Aggregate Service Times** The class-dependent service times can be reduced to a single aggregate service time. In Section (2.4.4) we have already introduced aggregate service times for OQNs. For CQNs we can define similar aggregate service times with the difference that the service times of each job class are weighed by their throughput instead of their arrival rate, see eq. (2.37) and (2.38). They are

\[
\mathbb{E}[S_j(\bar{n})] = \sum_{r=1}^{R} \frac{TH_j^{(r)}(\bar{n})}{TH_j(\bar{n})} \mathbb{E}_r S_j^{(r)} , \tag{3.34}
\]

\[
\mathbb{E}[S_j^2(\bar{n})] = \sum_{r=1}^{R} \frac{TH_j^{(r)}(\bar{n})}{TH_j(\bar{n})} \mathbb{E}_r \left(S_j^{(r)}\right)^2 . \tag{3.35}
\]

Note that the throughputs are state dependent, since the throughput may vary with each state \( \bar{n} \). Thus the aggregate service time, \( \mathbb{E}[S_j(\bar{n})] \), is also state dependent and has to be computed in each step of the algorithm.

With equations (3.34, 3.35) we can rewrite equation (3.33) as

\[
\mathbb{E}[W_j^{(r)}(\bar{n})] = Q_j(\bar{n} - \bar{e}_r) \mathbb{E}[S_{rem,j}(\bar{n})] + \mathbb{E}[L_{Q,j}(\bar{n} - \bar{e}_r)] \frac{\mathbb{E}[S_j(\bar{n} - \bar{e}_r)]}{e_j} \mathbb{E}[S_j^{(r)}] , \tag{3.36}
\]
where the remaining service time is

\[
\mathbb{E}S_{\text{rem},j}(\bar{n}) = \frac{c_j - 1}{c_j + 1} \frac{\mathbb{E}S_j(\bar{n} - \bar{e}_r)}{c_j} + \frac{2}{c_j + 1} \frac{1}{c_j} \frac{\mathbb{E}S_j^2(\bar{n} - \bar{e}_r)}{2\mathbb{E}S_j(\bar{n} - \bar{e}_r)}. \tag{3.37}
\]

Note that the generalized arrival theorem is applied by using the states \((\bar{n} - \bar{e}_r)\).

In the next paragraph we will find equations for \(Q_j(\bar{n})\) and \(\mathbb{E}L_{Q,j}(\bar{n})\).

**Reduced Complexity** From the multi-class MDA algorithm we can see that the complexity (number of multiplications) is determined by step 4 and step 6. The complexity is of the order

\[
MNR \prod_{r=1}^{R} (N^{(r)} + 1).
\]

The complexity can be reduced by using the variables \(Q_j(\bar{n})\) and \(\mathbb{E}L_{Q,j}(\bar{n})\). We can find the following recursive relations for both variables using equations (3.21, 3.26 and (3.32) and extending them for multiple classes

\[
Q_j(\bar{n}) = \frac{\mathbb{E}S_j(\bar{n})}{c_j} \sum_{r=1}^{R} TH_j^{(r)}(\bar{n}) [Q_j(\bar{n} - \bar{e}_r) + p_j(c_j - 1|\bar{n} - \bar{e}_r)], \tag{3.38}
\]

\[
\mathbb{E}L_{Q,j}(\bar{n}) = \frac{\mathbb{E}S_j(\bar{n})}{c_j} \sum_{r=1}^{R} TH_j^{(r)}(\bar{n}) [\mathbb{E}L_{Q,j}(\bar{n} - \bar{e}_r) + Q_j(\bar{n} - \bar{e}_r)]. \tag{3.39}
\]

From these equations we can see that the only marginal probabilities needed are \(p_j(c_j - 1|\bar{n} - \bar{e}_r), r = 1, \ldots, R\). Thus only the marginal probabilities up to having \(c_j - 1\) jobs at any station \(j\) need to be determined instead of all marginal probabilities. Hence the complexity of the algorithm is reduced to

\[
MRc^* \prod_{r=1}^{R} (N^{(r)} + 1),
\]

where \(c^* = \max\{c_1, \ldots, c_M\}\). Thus, if \(N\) is large, the complexity is considerably reduced.

Since not all marginal probabilities \(p_j(k|\bar{n}), k = 1, \ldots, n\) are known we need to formulate a new equation for the probability \(p_j(0|\bar{n})\). Using the variable \(Q_j(\bar{n})\), we obtain

\[
p_j(0|\bar{n}) = 1 - \sum_{k=1}^{c_j - 1} p_j(k|\bar{n}) - Q_j(\bar{n}). \tag{3.40}
\]

With the MVA algorithm in Section 3.1.3 we have seen that, to determine the expected time in the system for single server stations, we do not need the marginal probabilities. Thus, by distinguishing between single and multi server stations we can further reduce the computational effort of the algorithm.

**Ex. 3.2.1** Why is the computational effort reduced when we distinguish between single and multi server stations? But not the order of complexity?
Thus we need to find an equations for the expected time at a station for single server stations. Eq. (3.36) can be written as

\[
\mathbb{E}W_{j}^{(r)}(\bar{n}) = \mathbb{E}D_{\text{rem},j}(\bar{n}) + \mathbb{E}TCQ_{j}^{(r)}(\bar{n}) + \mathbb{E}S_{j}^{(r)},
\]

which leaves us to specify equations for \( \mathbb{E}D_{\text{rem},j}(\bar{n}) \) and \( \mathbb{E}TCQ_{j}^{(r)}(\bar{n}) \).

For single server stations the probability that the server is busy is given by

\[
\text{utilization of the station}
\]

which thus the expected delay until the first departure is given by

\[
\mathbb{E}D_{\text{rem},j}(\bar{n} - \bar{e}_r) = Q_j(\bar{n} - \bar{e}_r) \mathbb{E}S_{j}^{(r)}(\bar{n}) = T H_j(\bar{n} - \bar{e}_r) \mathbb{E}S_{j}^{(r)}(\bar{n} - \bar{e}_r) + \frac{\mathbb{E}S_j^2(\bar{n} - \bar{e}_r)}{2 \mathbb{E}S_j(\bar{n} - \bar{e}_r)},
\]

where we have used equation (3.37) for the remaining service time with \( c_j = 1 \).

The time to clear the queue for single server stations can be found from

\[
\mathbb{E}TCQ_j(\bar{n} - \bar{e}_r) = \sum_{s=1}^{R} [\mathbb{E}L_j^{(s)}(\bar{n} - \bar{e}_r) - T H_j^{(s)}(\bar{n} - \bar{e}_r) \mathbb{E}S_j^{(s)}] \mathbb{E}S_j^{(s)}^{(r)},
\]

where \( \mathbb{E}L_j^{(s)}(\bar{n} - \bar{e}_r) - T H_j^{(s)}(\bar{n} - \bar{e}_r) \mathbb{E}S_j^{(s)} \) denotes the average number of class-\( s \) jobs in the queue (see why!) and where \( \mathbb{E}L_j^{(s)}(\bar{n}) = T H_j^{(s)}(\bar{n}) \mathbb{E}W_j^{(s)}(\bar{n}) \).

We can now formulate a multi-class AMVA algorithm to approximate the performance measures of WCMS.

**Algorithm 6 Multi Class Approximate Mean Value Analysis**

1. (Initialization) For \( j = 0, \ldots, M \), if \( c_j > 1 \), set \( p_j(0 | \bar{0}) = 1 \), \( Q_j(\bar{0}) = 0 \), \( \mathbb{E}L_{Q,j}(\bar{0}) = 0 \), if \( c_j = 1 \), set \( \mathbb{E}L_{j}(\bar{0}) = 0 \).

2. For \( n = 1, \ldots, N \),
   - For all states \( \bar{n} \) for which \( \sum_{r=1}^{R} n^{(r)} = n \) and \( 0 \leq n^{(r)} \leq N^{(r)} \),
     - (a) For \( r = 1, \ldots, R \), and \( j = 0, \ldots, M \), if \( c_j > 1 \),
       \[
       \mathbb{E}W_j^{(r)}(\bar{n}) = Q_j(\bar{n} - \bar{e}_r) \mathbb{E}S_{j}^{(r)}(\bar{n}) + \mathbb{E}L_{Q,j}(\bar{n} - \bar{e}_r) \mathbb{E}S_j^{(r)} \mathbb{E}S_j^{(s)}(\bar{n} - \bar{e}_r) + \mathbb{E}S_j^{(r)}(\bar{n} - \bar{e}_r),
       \]
     - where
       \[
       \mathbb{E}S_{\text{rem},j}(\bar{n}) = \frac{c_j - 1}{c_j + 1} \mathbb{E}S_j(\bar{n} - \bar{e}_r) + \frac{1}{c_j + 1} \mathbb{E}S_j^2(\bar{n} - \bar{e}_r),
       \]
     - and if \( c_j = 1 \),
       \[
       \mathbb{E}W_j^{(r)}(\bar{n}) = \sum_{s=1}^{R} [\mathbb{E}L_j^{(s)}(\bar{n} - \bar{e}_r) - T H_j^{(s)}(\bar{n} - \bar{e}_r) \mathbb{E}S_j^{(s)}] \mathbb{E}S_j^{(s)} + T H_j(\bar{n} - \bar{e}_r) \mathbb{E}S_j(\bar{n} - \bar{e}_r) \mathbb{E}S_j^{(s)}(\bar{n} - \bar{e}_r) + \mathbb{E}S_j^{(s)}(\bar{n} - \bar{e}_r).
       \]
3.2. GENERAL WORKLOAD CONTROLLED MANUFACTURING SYSTEMS

(b) For \( r = 1, \ldots, R \) and \( j = 1, \ldots, M \),

\[
TH_0^{(r)}(\bar{n}) = \frac{n^{(r)}}{\sum_{i=0}^{M} V_i^{(r)} E W_i^{(r)}(\bar{n})},
\]

\[
TH_j^{(r)}(\bar{n}) = V_j^{(r)} TH^{(r)}(\bar{n}).
\]

(c) For \( j = 0, \ldots, M \),

\[
TH_j(\bar{n}) = \sum_{r=1}^{R} TH_j^{(r)}(\bar{n}),
\]

and if \( TH_j(\bar{n}) > 0 \),

\[
\mathbb{E} S_j(\bar{n}) = \sum_{r=1}^{R} \frac{TH_j^{(r)}(\bar{n})}{TH_j(\bar{n})} \mathbb{E} S_j^{(r)}, \quad \mathbb{E} S_j^2(\bar{n}) = \sum_{r=1}^{R} \frac{TH_j^{(r)}(\bar{n})}{TH_j(\bar{n})} \mathbb{E} \left( S_j^{(r)} \right)^2.
\]

(d) For \( j = 0, \ldots, M \), if \( c_j > 1 \) and \( n < c_j \) then \( Q_j(\bar{n}) = 0 \), if \( c_j > 1 \) and \( n \geq c_j \),

\[
Q_j(\bar{n}) = \frac{\mathbb{E} S_j(\bar{n})}{c_j} \sum_{r=1}^{R} TH_j^{(r)}(\bar{n}) \left[ Q_j(\bar{n} - \bar{e}_r) + p_j(c_j - 1 \mid \bar{n} - \bar{e}_r) \right].
\]

(e) For \( j = 0, \ldots, M \), and \( k = 1, \ldots, c_j - 1 \), if \( c_j > 1 \),

\[
k \frac{1}{\mathbb{E} S_j(\bar{n})} p_j(k \mid \bar{n}) = \sum_{r=1}^{R} TH_j^{(r)}(\bar{n}) p_j(k - 1 \mid \bar{n} - \bar{e}_r),
\]

\[
p_j(0 \mid \bar{n}) = 1 - \sum_{l=1}^{c_j - 1} p_j(k \mid \bar{n}) - Q_j(\bar{n}).
\]

(f) For \( j = 0, \ldots, M \), if \( c_j > 1 \),

\[
\mathbb{E} L_{Q,j}(\bar{n}) = \frac{\mathbb{E} S_j(\bar{n})}{c_j} \sum_{r=1}^{R} TH_j^{(r)}(\bar{n}) \left[ \mathbb{E} L_{Q,j}(\bar{n} - \bar{e}_r) + Q_j(\bar{n} - \bar{e}_r) \right].
\]

(g) For \( r = 1, \ldots, R \), and \( j = 0, \ldots, M \), if \( c_j = 1 \),

\[
\mathbb{E} L_j^{(r)}(\bar{n}) = TH_j^{(r)}(\bar{n}) E W_j^{(r)}(\bar{n}).
\]
Chapter 4

PAC systems

4.1 Introduction

So far, we have studied manufacturing systems that are modelled as either open queueing networks (OQN’s) or networks with a maximum number of jobs (CQN’s or ROQN’s). The number of jobs in an open network may vary significantly due to randomness in the arrival processes, service times and job routings. In order to keep the work on the shop floor manageable, and to restrict the internal cycle times, shop floor managers often put a limit on the number of jobs at the shop floor. In principle, this can be modeled as a closed or restricted open network. However, in the preceding chapter we simply ignored any jobs that could not entry the shop floor immediately due to a lack of available pallets or tokens (the latter often serve as a production authorization card or PAC). In this chapter we no longer ignore externally waiting jobs. In reality, production authorization cards often serve to control the release of work to the shop floor. Hence, as long as there is no authorization for a job release, the relevant material is kept in an intermediate store. In this chapter, we study both production to order, and production to stock systems in which the number of jobs released to the shop floor is controlled by a fixed number of production authorization cards (PAC’s).

4.1.1 Production to order PAC systems

Here we will discuss the single class production to order PAC system, in which jobs arrive according to some random process. Within the production system, a finite number of production authorization cards (PAC’s) are available. A job may only start execution if a free PAC can be attached to it (hence the name of the card). Upon completion of the job, it leaves the system to fulfill the customer’s demand while the attached card is placed in a card box, or immediately attached to a next job if all cards are occupied.

Assume that jobs arrive according to a Poisson arrival process with rate \( \lambda \). The main part of the model is the network which consists of \( M \), possibly multi-server, stations, modeling the workstations. At station \( j \), jobs receive service according to some general service time distribution with first two moments \( E[S_j] \) and \( E[S_j^2] \), while a job’s routing across the workstations is determined by the routing matrix \( P \). Figure 4.1 displays a PAC based production to order system with
$N$ cards. In this figure, we observe an external so-called synchronization station in which $a$ denotes the number of jobs waiting for a card, while $b$ denotes the number of free cards. Obviously, one of these two queues is always empty. This station operates similarly to the load/unload station discussed earlier, except that externally waiting jobs are no longer ignored. The synchronization queue station also bears some similarity with a similar construction in a production to stock system, but there is a clear difference regarding the output of the station. Under a PAC regime, a customer order is not necessarily immediately transferred to the shop floor (contrary to the production to stock models discussed so far), since a constraint is put on the amount of work in process. This makes the analysis of such systems essentially more difficult. On the other hand, a completed job leaves the system immediately while the attached card is ready for use to a next job, if any.

In this system, the routing probabilities are given by

$$
\begin{align*}
P_{0i} &= \text{the probability that the first operation is at station } i, \\
P_{ij} &= \text{the probability that upon completion at station } i \\
&\quad \text{the next operation is at station } j, \\
P_{j0} &= \text{the probability that upon completion at station } j \\
&\quad \text{the job leaves the system.}
\end{align*}
$$

We assume that the system is stable. That is, we assume that there is both sufficient workstation capacity and there is a sufficient number of cards to process the offered work. More precisely, we assume that the service times and the arrival rates are such that without the PAC constraint the workstations have sufficient capacity to process all arriving jobs:

$$
\frac{\lambda_j E S_j}{c_j} = \rho_j < 1, \quad j = 1, \ldots, M
$$

where $\lambda_j$ denotes the arrival rate at station $j$, $c_j$ the number of servers at station $j$, and $\rho_j$ the total utilization of station $j$. The value of $\lambda_j$ follows from the external arrival rate and the routing matrix according to

$$
\lambda_j = \lambda P_{0j} + \sum_{i=1}^{M} \lambda_i P_{ij}, \quad j = 1, \ldots, M.
$$
4.2. EXACT ANALYSIS

The visit ratios can be obtained from

\[ V_j = \frac{\lambda_j}{\lambda}, \quad j = 1, \ldots, M. \]

This is also obtained by solving the homogenous equation \( \bar{V} = \bar{V} \hat{P} \), as we have done previously with CQNs, and setting the visit ratio of the synchronization station, denoted by index \( M + 1 \), equal to 1 (jobs synchronize only once with a PAC), where the routing matrix \( \hat{P} \) is defined by

\[ \hat{P}_{ij} = P_{ij}, \quad i, j = 1, \ldots, M, \]
\[ \hat{P}_{i,M+1} = P_{i0}, \quad i = 1, \ldots, M, \]
\[ \hat{P}_{M+1,j} = P_{0j}, \quad j = 1, \ldots, M. \]

Furthermore, given that the workstations have sufficient capacity, we assume that there are a sufficient number of cards in the system (i.e. \( N \) is sufficiently large) to prevent the external queue from tending to infinity. This assumption is somewhat more difficult to formalize, but we will do so in Section 4.2.3 for a PAC based production to order system with exponential service times.

4.1.2 Overview of this chapter

In this chapter, we discuss the exact and approximate analysis of the single-class PAC production to order system, and an approximate analysis of multi-class systems. Typical performance measures that we are interested in are the mean flow time including the time a job waits to enter the network, and the mean number of jobs waiting in the external queue. In Section 4.2, we present an exact analysis of the SOQN-EQ with exponential service times. The exact analysis is carried out using the matrix-geometric approach as described by Neuts [60]. It provides performance measures and relates the stability condition of the PAC system to a constructed CQN. Next an approximate method, introduced by Buitenhek [15], is presented to evaluate the general single class PAC production to order system. Next, we discuss PAC production to stock systems in which a base stock of finished products is held in stock to immediately fill customer demand. In Sections 4.5 and 4.6 multi-class PAC systems are discussed. Here, we distinguish systems in which there is a general set of cards for all job types, and systems in which the number of jobs of each class is limited by a dedicated card set.

4.2 Exact analysis

For the single-class SOQN-EQ with exponential service times, we can in principle provide an exact analysis. We refer to SOQN-EQ with exponential service times as to exponential SOQN-EQ.

4.2.1 The Markov process

The behavior of the exponential SOQN-EQ is described by a Markov process with states

\[ (n_E, \bar{n}_I) = (n_E, n_1, \ldots, n_M), \]
where \( n_E \) is the number of jobs in the external queue and \( \tilde{n}_I = (n_1, \ldots, n_M) \) is the population of jobs inside the network. The number of jobs inside the network is defined as \( n_I = |\tilde{n}_I| = \sum_{j=1}^{M} n_j \).

Let \( n_E > 0 \). Then, from a state \((n_E, n_1, \ldots, n_M)\) of this Markov process, there can be transitions due to a job arrival, or due to a departure at some station. An arrival leads to a transition to the state \((n_E + 1, n_1, \ldots, n_M)\). A departing job at station \( i \) may either leave the system or move to another station, say station \( j > i \). If the job leaves the system, then it is replaced by the first job in the external queue. Let this job have its first operation at station \( j > i \), as well. Then, these events lead to the transitions to the states \((n_E - 1, n_1, \ldots, n_i - 1, \ldots, n_j + 1, \ldots, n_M)\) and \((n_E, n_1, \ldots, n_i - 1, \ldots, n_j + 1, \ldots, n_M)\) respectively. For \( n_E = 0 \), the transitions are different, but very similar. Hence, this Markov process is a quasi birth-death process, since there can only be transitions from states with \( n_E = \tilde{n}_E \) to states with \( n_E = \tilde{n}_E - 1, n_E = \tilde{n}_E, \) or \( n_E = \tilde{n}_E + 1 \).

**Example 4.1**

Consider the two-machine flowshop (i.e., \( M = 2, c_j = 1 \) for \( j = 1, 2, P_{01} = P_{12} = P_{20} = 1 \)) with \( N = 3 \) pallets. The state-transition diagram of the Markov process for this SOQN-EQ has been depicted in Figure 4.2. In this Markov process, the transitions to the east are with rate \( \lambda \), the transitions to the north are with rate \( \mu_1 \) and the transitions to the south-west are with rate \( \mu_2 \). This Markov process is a quasi birth-death process.

![Figure 4.2: The state-transition diagram of the Markov process describing the behavior of the exponential single-class SOQN-EQ with M=2 and N=3.](image-url)
4.2. EXACT ANALYSIS

The generator $Q$ has a special structure, related to the quasi birth-death process character. The state space of the Markov process consists of levels, $k = 0, 1, 2, \ldots$, where each level $k$ is defined as the set of states with $n_k = k$. For all $k > 0$, the number of states $L$ is given by the number of population states $(n_1, \ldots, n_M)$ for which $\sum_{j=1}^M n_j = N$. Hence, we find $L = \binom{N+M-1}{N}$.

The number of states in level 0 is $L_0 = \sum_{n=0}^N \binom{n+M-1}{n}$. For the flowshop of Example 4.1, $L = 4$ and $L_0 = 10$. Let the states within each level $k \geq 1$ be in reversed lexicographical order. For the flowshop of Example 4.1 this means that the states at level $k$ are ordered as $(k, 0, N), (k, 1, N-1), \ldots, (k, N, 0)$. The generator $Q$ of the Markov process may be written as

$$Q = \begin{pmatrix} B_{00} & B_{01} & 0 \\ B_{10} & A_1 & A_0 \\ 0 & A_2 & A_1 \\ 0 & 0 & 0 \end{pmatrix}$$

where the submatrices $A_1, B_{00}, B_{01},$ and $B_{10}$ are as follows. Each matrix $A_i \in \mathbb{R}^{L \times L}$ contains the rates for the transitions from level $k$ to level $k+1-i$. The matrix $B_{00} \in \mathbb{R}^{L_0 \times L_0}$ contains the rates for the transitions inside level 0. The matrix $B_{01} \in \mathbb{R}^{L_0 \times L}$ contains the rates for the transitions from level 0 to level 1. The matrix $B_{10} \in \mathbb{R}^{L \times L_0}$ contains the rates for the transition from level 1 to level 0. For the flowshop of Example 4.1, we obtain

$$A_0 = \lambda I_4$$

$$A_1 = \begin{pmatrix} -(\lambda + \mu_1) & \mu_1 & 0 & 0 \\ 0 & -(\lambda + \mu_1 + \mu_2) & \mu_1 & 0 \\ 0 & 0 & -(\lambda + \mu_1 + \mu_2) & \mu_1 \\ 0 & 0 & 0 & -(\lambda + \mu_2) \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \mu_2 & 0 & 0 & 0 \\ 0 & \mu_2 & 0 & 0 \\ 0 & 0 & \mu_2 & 0 \end{pmatrix}$$

where $I_4$ denotes the 4-dimensional unit matrix. Level 0 may be partitioned into sublevels consisting of states with the same total number of jobs (or equivalently, the same number of jobs inside the network). Assume that the states of level 0 are ordered according to these sublevels and that the states within each sublevel are ordered in the same way as for the levels $k \geq 1$. Then $B_{01} = (0 \mid A_0^T)^T$ and $B_{10} = (0 \mid A_2)$.

4.2.2 The stationary distribution

It can be shown that the stationary distribution of the continuous-time Markov process with generator $Q$ given above equals the stationary distribution of the discrete time Markov process with the same state space and with transition matrix $P$ given by
in which \( \hat{A}_0 = \gamma A_0, \hat{A}_1 = I + \gamma A_1, \hat{A}_2 = \gamma A_2 \) and \( \gamma \) is the reciprocal of largest element in absolute value on the diagonal of \( A_1 \). Of course, the stationary distribution \( \pi \) can only be obtained if the Markov process is stable. For the time being, we assume that this is the case. In Section 4.2.3, the stability condition for this Markov process is discussed in detail.

For \( k > 0 \), let \( \bar{\pi}_k = (\pi_{k,1}, \ldots, \pi_{k,L}) \), where \( \pi_{k,i} \) denotes the stationary probability of the \( i \)-th state of level \( k \) in the lexicographical ordering, and let \( \bar{\pi}_{0,N} = (\pi_{0,L_0-L+1}, \ldots, \pi_{0,L_0}) \). It follows from the matrix-geometric theory that a non-negative matrix \( R \in \mathbb{R}^{L \times L} \) exists for which

\[
\bar{\pi}_k = \bar{\pi}_{0,N} R^k, \quad k \geq 1,
\]

and \( R \) is the solution of the quadratic matrix equation

\[
R = \hat{A}_0 + R \hat{A}_1 + R^2 \hat{A}_2.
\]

From this relation, the matrix \( R \) can be obtained by successive substitution (starting with \( R^{(0)} = 0 \)) or more efficiently by the improved solution procedure as described by Latouche and Ramaswami [46]. In addition, it can be shown that the matrix \((I - R)^{-1}\) exists.

From Equation (4.2), we can derive equations for the mean number of jobs in the external queue \( \mathbb{E}L_E \) and the mean number of jobs in the network \( \mathbb{E}L_I \). We first introduce some extra notation. Let \( e \) denote the column vector of appropriate size of which all elements are equal to one. Denote by \( \bar{n}_0 \) the vector of which the components \( n_{0,i} \) give the number of jobs in the network for the \( i \)-th state in level \( 0 \); and by \( \bar{\pi}_0^T \) the transposed of \( \bar{\pi}_0 \). We find,

\[
\mathbb{E}L_E = \sum_{k=1}^{\infty} k \bar{\pi}_k e = \sum_{k=1}^{\infty} k \bar{\pi}_{0,N} e = \bar{\pi}_{0,N} R(I - R)^{-2} e,
\]

\[
\mathbb{E}L_I = \bar{n}_0 \bar{\pi}_0^T + N \sum_{k=1}^{\infty} \sum_{i=1}^L \pi_{k,i} = \bar{n}_0 \bar{\pi}_0^T + N \sum_{k=1}^{\infty} \bar{\pi}_k e
\]

\[
= \bar{n}_0 \bar{\pi}_0^T + N \sum_{k=1}^{\infty} \bar{\pi}_0 R^k e = \bar{n}_0 \bar{\pi}_0^T + N \bar{\pi}_0 \sum_{k=1}^{\infty} R^k e
\]

\[
= \bar{n}_0 \bar{\pi}_0^T + N \bar{\pi}_0 R(I - R)^{-1} e.
\]

These expressions contain unknown stationary probabilities for states in level \( 0 \). These probabilities are obtained by solving the balance equations for the states level \( 0 \), while using the relation (4.2) for \( k = 1 \), and the normalization condition, which states that the sum of all probabilities is equal to one.

In this way, the desired mean queue lengths are obtained. Using Little’s law the mean waiting times in the external queue and the cycle times in the network are obtained and, hence, the mean total flow time. Obviously, one may obtain the entire stationary distribution of the Markov process in a similar way.
4.2. EXACT ANALYSIS

Figure 4.3: The CQN constructed for the intuitive explanation of the ergodicity condition and for Step 1 of the Aggregation method.

4.2.3 The stability condition

The stationary distribution of the Markov process only exists if the Markov process is stable. We formulate the stability condition. Let $\tilde{A} = \sum_{i=0}^{2} A_i$. It is easily verified that $\tilde{A}$ is a stochastic matrix. Denote by $\tilde{\pi}$ the stationary distribution of the Markov process associated with $\tilde{A}$. In order for the Markov process associated with $Q$ (or $P$) to be stable, it must hold that

$$\tilde{\pi} \tilde{A}_0 e < \tilde{\pi} \tilde{A}_2 e.$$

(4.4)

Inequality (4.4) states that at high levels of the Markov process associated with $Q$, the equilibrium probability flow to the right (more jobs in the external queue) should be smaller than the equilibrium probability flow to the left (less jobs in the external queue).

Furthermore, an intuitive inequality can be obtained from Inequality (4.4). In particular, the condition may be shown to relate to a CQN which is constructed from the PAC based production to order system as follows. This CQN consists of stations 1 to $M$ (with identical service times and service disciplines as in the PAC system), has a fixed number of $N$ jobs, and routing probabilities $\tilde{P}_{ij}$ given by

$$\tilde{P}_{ij} = P_{ij} + P_{i0} P_{0j}, \quad i, j = 1, \ldots, M.$$  

(4.5)

The visit ratios $\tilde{V}_j$ of this CQN can be chosen such that $\tilde{V}_j = V_j$, $j = 1, \ldots, M$. The resulting network is illustrated in Figure 4.3. In this CQN, the link created to obtain the CQN (and replacing the external arrivals and departures of jobs) is called the card feedback loop. Denote by $TH(N)$ the throughput of the CQN with $N$ jobs as measured at the card feedback loop (if $\tilde{V}_j = \tilde{V}_j$). Then, by substituting $\tilde{\pi}$, $\tilde{A}_0$, and $\tilde{A}_2$, for this particular model, Inequality (4.4) can be rewritten as

$$\lambda < TH(N).$$

(4.6)

Intuitively, this result is clear: the maximum throughput of the PAC system is obtained when all cards are always in use. This is achieved in the described CQN. (Alternatively, this CQN can be seen as the CQN obtained in which cards simply skip the synchronization station.) From this intuitive reasoning, it also follows immediately that this result also holds for the PAC system with general service times. We illustrate it for the exponential PAC production to order system in Example 4.2. By verifying Inequality (4.4) for several values of $N$, one can obtain the minimum number of cards required to meet demand in the PAC based production to order system.
Example 4.2  
Reconsider the 2-machine flowshop of Example 4.1. For this system, we find

\[ \hat{A} = \frac{1}{\lambda + \mu_1 + \mu_2} \begin{pmatrix} \lambda + \mu_2 & \mu_1 & 0 & 0 \\ \mu_2 & \lambda & \mu_1 & 0 \\ 0 & \mu_2 & \lambda & \mu_1 \\ 0 & 0 & \mu_2 & \lambda + \mu_1 \end{pmatrix} . \]

The stationary distribution corresponding to the probability matrix \( \hat{A} \) equals the one corresponding to generator \( \hat{A} \) given by

\[ \hat{A} = \begin{pmatrix} -\mu_1 & 0 & 0 & \mu_1 \\ 0 & -\mu_2 & 0 & \mu_1 \\ 0 & \mu_2 & -(\mu_1 + \mu_2) & \mu_1 \\ 0 & 0 & \mu_2 & -\mu_2 \end{pmatrix} , \]

which is the generator of the CQN with routing probabilities \( P_{12} = P_{21} = 1 \). Therefore, \( \hat{\pi} \) is the stationary distribution of this CQN. Let us now consider the right-hand side of Inequality (4.4). This yields

\[ \hat{\pi} \hat{A}_2 e = \sum_{i=2}^{L} \hat{\pi}_i \mu_2 = \text{Prob( station 2 is busy )} \mu_2 = TH_2(N) , \]

where \( TH_2(N) \) denotes the throughput at station 2. The card feedback loop in this CQN is the link from station 2 to station 1. Since \( P_{21} = 1 \), the throughput \( TH(N) \) as measured at the card feedback loop equals the throughput at station 2, i.e., \( TH(N) = TH_2(N) \). In other words, the stability condition is formulated as \( \lambda < TH(N) \).

4.3 Approximating production to order models

A first aggregation method to evaluate single class PAC production to order systems has been presented by Avi-Itzhak and Heyman [4], and Buzacott and Shanthikumar [21]. Here we discuss an equivalent method: the CQN-view based aggregation method as introduced by Buitenhek [15]. The method starts with the transformation of the PAC production to order system into a CQN including the synchronization station (see Figure 4.4). Identify in this CQN the synchronization station with the index \( M+1 \). We refer to this CQN as the CQN-view of the PAC production to order system. To obtain the CQN-view, the roles of the jobs and the cards (PAC’s) are interchanged. The cards are now the main entities and they are delayed at the synchronization station, because they need a job before entering a new cycle.

Given the CQN-view of the PAC system, the aggregation method proceeds by evaluating the CQN without the synchronization station to determine the load-dependent throughput \( TH(n) \) of the initial network. Based upon these throughputs, next the synchronization station is analyzed in isolation to determine the expect number of jobs in the external queue and the expected waiting time. This analysis in turn is used to obtain the CQN view, by replacing the synchronization queue by a load-dependent exponential server with station index \( M+1 \), which describes the behavior of the node as experienced by a PAC.
4.3. APPROXIMATING PRODUCTION TO ORDER MODELS

The initial network together with the load-dependent server is denoted as the CQN-LD. The performance measures concerning the workstations are now obtained by analyzing the CQN-LD using for example an (approximate) MVA algorithm.

In the following subsections we will describe these four steps in more detail:

1. Evaluation of the CQN without the synchronization station
2. Construction of a load-dependent server as a substitute for the synchronization station.
3. Evaluation of the CQN-LD including the synchronization station

4.3.1 Evaluation of the CQN without the synchronization station

The main objective of the evaluation of the CQN without the synchronization station is to determine the load-dependent throughput $TH(n)$, which we need in the next step (there we will discuss why) and for the evaluation of the synchronization station in isolation.

To evaluate exponential CQNs we can use Norton’s theorem, which was formulated by Chandy, Herzog and Woo [25]. In principle, Norton’s theorem decomposes a network with $M$ stations into an equivalent network with 2 stations. One station is identical to one in the original network and the other station is a load-dependent exponential server that replaces the rest of the original network. An important concept that facilitates the formulation of Norton’s theorem is the complement (network) of a queue. The complement of station $i$ in a CQN is constructed from the original network by setting the service time at station $i$ to zero. An alternative view on the complement of station $i$ is a network consisting of all stations of the original CQN except station $i$, in which the jobs originally arriving at station $i$ are now immediately redirected according to their routing. The empty place replacing station $i$ is called the short.

Norton’s theorem for single-class queueing networks is formulated as follows.

**Theorem 4.3** (Norton’s theorem for queueing networks, Chandy, Herzog and Woo [25]) Let $C$ be a GN-network. Let the equivalent network $C_{eq}$ consist of any
station $i$ of the original network and a composite queue with load-dependent exponential service rates $\mu_{eq}(n)$ equal to the throughput $TH(n)$ of the complement network of station $i$ with $n$ jobs, as measured at the short (for $n = 0, 1, \ldots, N$).

Then the queue length distribution of station $i$ in the equivalent network is the same as in the original network:

$$ p_{eq,i}(n) = p_i(n) \text{ for } n = 0, 1, \ldots, N, \quad (4.7) $$

where $p_i(n)$ denotes the stationary probability that there are $n$ jobs at station $i$ in $C$, and $p_{eq,i}(n)$ denotes the stationary probability that there are $n$ jobs at station $i$ in $C_{eq}$.

Note that $C$ should be a GN-network and thus the service times should be exponentially distributed. To be able to approximate the load-dependent throughput of non-product form CQNs we will assume that Norton’s theorem still holds.

To obtain the complement network of station $M+1$, denoted with the superscript $C$, the routing matrix is adjusted according to

$$ P_{ij}^C = P_{ij} + P_{i,M+1} P_{M+1,j}, \quad (4.8) $$

such that the jobs bypass the synchronization station.

Now we can use Algorithm 5 to evaluate the complement network.

### 4.3.2 Constructing a load-dependent server as a substitute for the synchronization station

Let us now take the CQN view and replace the synchronization station by a load-dependent server. Based upon the application of Norton’s theorem, the arrival rates of cards at this load-dependent server follow from the throughputs of the complement network (see below). First however, we describe how to obtain the service rates $\mu_{M+1}(n)$. Consider the arrival of a card at the synchronization station. If the arriving card observes already other cards waiting at the synchronization station, then the remaining time until the next departure of a card (that starts its manufacturing cycle) equals the time remaining until the next arrival of a job. Therefore, it is logical to set $\mu_{M+1}(n) = \lambda$, for $n = 2, \ldots, N$. However, if the arriving card observes no other cards waiting, there is a probability $q$, say, that the card still has to wait because also the job queue is empty, whereas with a probability $1 - q$ a job is waiting and the card may proceed immediately (hence has a zero service time). Consequently, the mean waiting time of an card at the load dependent station with no other cards present equals $q/\lambda$; therefore it seems natural to choose $\mu_{M+1}(1) = \lambda/q$. It remains to estimate $q$. To this end, consider the analysis of the synchronization station in isolation. Describe the state of the synchronization station by the tuple $(n_P, n_J)$, where $n_P$ and $n_J$ denote the number of pallets and the number of jobs at the synchronization station, respectively. We know that the jobs arrive from outside according to a Poisson process with rate $\lambda$. Furthermore, from Norton’s theorem we find the load-dependent arrival rate $\lambda_{M+1}(n)$ of cards at the synchronization station if there are $n$ cards present as the throughput of the complement network of the synchronization station if there are $N - n$ pallets in it: $\lambda_{M+1}(n) = TH_{C_{M+1}}(N - n)$. The behavior of the synchronization station
is now described by a simple birth-death process. The feasible states of this
process are given by \((n_P, 0), 0 \leq n_P \leq N\) and \((0, n_J), n_J > 0\).

\[
\begin{align*}
\lambda_{M+1}(0) & \quad \lambda_{M+1}(1) & \quad \lambda_{M+1}(2) \\
0,0 & \quad 1,0 & \quad 2,0 & \quad 3,0 \\
\lambda_{M+1}(0) & \quad \lambda & \quad \lambda & \quad \lambda \\
0,1 & \quad 0,2 & \quad 0,3 \\
\lambda_{M+1}(0) & \quad \lambda \\
\lambda_{M+1}(0) & \quad \lambda
\end{align*}
\]

Figure 4.5: The birth-death process describing the behavior of the synchroni-
station when \(N = 3\).

The state-transition diagram for the birth-death process is given in Figure
4.5. The balance equations for this process are given by

\[
\begin{align*}
\lambda_{M+1}(n) & = \mathcal{H}_{M+1}^C(N - n_P) p(n_P - 1, 0), \quad n_P = 1, \ldots, N, \quad (4.9) \\
\mathcal{H}_{M+1}^C(N) p(0, n_J) & = \lambda p(0, n_J - 1), \quad n_J > 0. \quad (4.10)
\end{align*}
\]

From this Markov process, we can obtain the probabilities \(p(n_P, n_J)\), and from
these an approximation of the probability \(q\) that there are no jobs given that
there are no cards at the synchronization station, as follows:

\[
q = \frac{p(0, 0)}{\sum_{n_J=0}^{\infty} p(0, n_J)}. \quad (4.11)
\]

From the balance equations, we obtain

\[
\sum_{n_J=0}^{\infty} p(0, n_J) = p(0, 0) \frac{1}{1 - \mathcal{H}_{M+1}^C(N)}. \quad (4.12)
\]

Substituting (4.12) into (4.11), we find

\[
q = 1 - \frac{\lambda}{\mathcal{H}_{M+1}^C(N)}.
\]

hence

\[
\mu_{M+1}(1) = \left[ \frac{1}{\lambda} - \frac{1}{\mathcal{H}_{M+1}^C(N)} \right]^{-1}. \quad (4.13)
\]

Note that also other relevant performance measures such as the mean number
of jobs in the external queue are immediately obtained from the above analysis.
An easy check on the consistency of the choice for \( q \) is made through the following observation. Since the original system is basically an open system with average input rate \( \lambda \), and every job passes the synchronization station only once, the throughput of this station, and hence of the load-dependent server should equal \( \lambda \) as well. Denote by \( p_{M+1}(0) \) and \( p_{M+1}(1) \) the marginal probabilities that there are 0 and 1 cards at the load-dependent server. Then, from the well-known local balance equations we have

\[
TH^{CM+1}(N)p_{M+1}(0) = \mu_{M+1}(1)p_{M+1}(1) \quad (4.14)
\]

Substituting 4.13 in 4.14 and rearranging terms yields:

\[
TH^{CM+1}(N)p_{M+1}(0) = \lambda[p_{M+1}(0) + p_{M+1}(1)] \quad (4.15)
\]

On the other hand, the throughput \( TH \) at the load-dependent server obviously equals:

\[
TH = p_{M+1}(1)\mu_{M+1}(1) + [1 - p_{M+1}(0) - p_{M+1}(1)]\lambda \quad (4.16)
\]

By combining (4.14), (4.15) and (4.16), we immediately obtain

\[
TH = \lambda,
\]

which shows the consistency of the choice of \( \mu_{M+1}(1) \).

If there are non-exponential servers we still choose the service rate \( \mu_{M+1}(n) \) of the synchronization station in the same way. In other words, we pretend that for this non-product-form network Norton’s theorem still holds.

### 4.3.3 Evaluation of the CQN-LD including the synchronization station

Let us consider the AMVA algorithm required to analyze the CQN-LD. In addition to the normal equations for the \( M \) workstations, we need equations for the load-dependent exponential server which replaces the synchronization station.

The expected time at an \( M/M(n)/1 \) queue for an arriving card is

\[
EW(n) = n\sum_{k=0}^{n-1} \frac{k+1}{\mu(k+1)}p(k|n-1)
\]

From this equation, \( \mu_{M+1}(1) = q/\lambda \) and \( \mu_{M+1}(n) = \lambda \), for \( n > 1 \), we find

\[
EW_{M+1}(n) = n\sum_{k=1}^{n-1} \frac{k+1}{\lambda}p_{M+1}(k|n-1) + p_{M+1}(0|n-1) \frac{q}{\lambda}. \quad (4.17)
\]

If we define (in accordance with similar definitions in Section 3.1.4) \( Q_{M+1}(n) \) as the probability that station \( M+1 \) is busy and \( EL_{M+1}(n) \) as the mean number of cards present at station \( M+1 \), we have

\[
Q_{M+1}(n) = \sum_{k=1}^{n} p_{M+1}(k|n),
\]

\[
EL_{M+1}(n) = \sum_{k=1}^{n} kp_{M+1}(k|n),
\]
and then Equation (4.17) can be written as

$$E_{W_{M+1}}(n) = [E_{L_{M+1}}(n - 1) + Q_{M+1}(n - 1)] \frac{1}{\lambda} + p_{M+1}(0 | n - 1) \frac{q}{\lambda}. \quad (4.18)$$

This equation shows the relation between the load-dependent server and the synchronization station which it replaces. At the synchronization station a card waits for the cards in front of him, and if there are no other cards, then with probability $q$ it waits $1/\lambda$ time units on average until the external arrival of an external job.

The AMVA algorithm then also needs recursive expressions for these variables, and for the probability $p_{M+1}(0 | n)$. These are obtained by straightforward algebra and extension of the equations for the other stations, and are formulated as

$$Q_{M+1}(n) = \frac{TH_{M+1}(n)}{\lambda} [Q_{M+1}(n - 1) + p_{M+1}(0 | n - 1) q], \quad (4.19)$$

$$p_{M+1}(0 | n) = 1 - Q_{M+1}(n), \quad (4.20)$$

$$E_{L_{M+1}}(n) = TH_{M+1}(n) E_{W_{M+1}}(n). \quad (4.21)$$

### 4.3.4 The Algorithm

With the steps described above we can now formulate a more formal algorithm to evaluate a single class PAC production to order system.

**Algorithm 7** Single Class PAC production to order system

1. Obtain the CQN-view of the PAC system.

2. Compute the throughputs $TH_{C_{M+1}}^C(n)$ of the complement network of station $M + 1$, $n = 1, \ldots, N$.

3. Replace the synchronization station by a load-dependent exponential server with service rates $\mu_{M+1}(n) = \lambda$ for $n = 2, \ldots, N$, and $\mu_{M+1}(1) = \lambda/q$, where $q = 1 - \lambda / TH_{C_{M+1}}^C(N)$.

4. Analyze the resulting CQN-LD (including station $M + 1$) to obtain the mean number of jobs at each station and the mean number of cards at station $M + 1$.

5. Obtain the performance measures concerning the external queue from the birth-death process, describing the behavior of the synchronization station.

### 4.4 Production to stock models

In Section 2.3 we have evaluated open systems in which a target inventory level of end items was assumed, hence items were produced in anticipation of future demand. The same can be done for PAC based systems but the analysis of such systems becomes a bit more complicated. In such systems, we have two regulating parameters, i.e. a base stock level $S$ and a number of PAC’s, denoted by $N$. Figure 4.6 displays the architecture of the system.
We assume that customers arrive according to a Poisson process with rate $\lambda$. Any arriving demand is split into two requests. One request ($k$) is directed to the stock ($m$) of finished products, at synchronization station $J_k$. If a product is available in stock, it is used to satisfy demand, otherwise this request joins a queue of waiting customers. The other request reflects a production order ($a$) for a similar item in order to make sure that the inventory position of finished items is kept on the same level. The production order will be released if and only if it is authorized by a free PAC from the queue of free cards ($b$), at synchronization station $J_b$. If no such card is available, the production order joins a queue of orders that are waiting for a PAC and hence still have to be released. Finally, in Figure 4.6, the vector $\vec{n}$ denotes the items present at each individual workstation in the manufacturing network.

From the above description, we easily deduce

\begin{align*}
|\vec{n}| + a + m - k &= S \\
|\vec{n}| + b &= N \\
m.k &= 0 \\
b.a &= 0
\end{align*}

From equations (4.22) and (4.23) we immediately deduce

\begin{equation}
(b - a) - (m - k) = N - S
\end{equation}

The reader should note that the two synchronization stations together constitute a one-dimensional system, since the underlying birth-death processes are identical. This is easily seen as follows. Consider first an initial state in which $b = N$ and $m = S$. There are only two events that can change the state of the two synchronization stations, i.e. an arrival of a customer that decreases both $m - k$ and $b - a$ by one, and the completion of a job that increases both $m - k$ and $b - a$ by one. Now, let us first consider the case $N < S$. Then it is
easy to see that \( m \geq b \) and, more specifically, that \( m = b + (S - N) \) as long as \( b > 0 \). That is: as long as the system is immediately responsive, the stock of finished items cannot drop below the level \( (S - N) \), and if \( m < S - N \), the system becomes less responsive, i.e. the release of production orders is delayed due to a lack of free PAC’s.

For the remainder of this section we therefore assume that \( N \geq S \). Then it is easy to see from equation (4.24) that

\[
\begin{align*}
  b &= \max [0, m + (N - S)], \\
  a &= \max [0, k - (N - S)].
\end{align*}
\]

The analysis of the system now proceeds as follows. Recall that the two synchronization stations together constitute a one-dimensional system. Hence, we start with analyzing the original manufacturing network for any number of jobs \( |\pi| \) with \( 0 \leq |\pi| \leq N \). The resulting throughput rates \( TH(N) \) are the arrival rates (of finished products and no longer occupied PAC’s) to queues \( m \) and \( b \) of the synchronization stations. The queues \( k \) and \( a \) are fed by the customer arrival rates. From this, it is easy to compute e.g. the probability distribution \( \pi(m - k) \) at station station \( J_k \) from the birth-death equations. For ease of notation, let \( b_m = m - k \). Then

\[
\begin{align*}
  \lambda \pi(\hat{m}) &= TH(S - \hat{m} + 1) \pi(\hat{m} - 1), \quad \text{for } \hat{m} = S, S - 1, \ldots, S - N + 1, \\
  \lambda \pi(\hat{m}) &= TH(N) \pi(\hat{m} - 1), \quad \text{for } \hat{m} = S - N, S - N - 1, \ldots.
\end{align*}
\]

The customer order fill rate \( FR \) then immediately follows from

\[
FR = \sum_{\hat{m}=1}^{S} \pi(\hat{m})
\]

On order to obtain other performance measures such as the mean number of jobs at each workstation and the mean response time of the system, we analyze the closed queueing network, consisting of the manufacturing system together with synchronization station \( J_b \), by the methods discussed in the preceding section (hence \( J_b \) is then replaced by a load-dependent server again).

Finally, note that a number of earlier discussed models are special cases of the one discussed here. For instance, if \( S = 0 \), we obtain the PAC-based production to order model again. If \( N = \infty \), we are back to the open production to order model discussed in Section 2.3.

### 4.5 Multi-Class Networks: universal cards

In this section we consider a PAC-based system that produces multiple part types, to order. These systems can be distinguished into systems that share a common set of production authorization cards for all classes, and systems that have a separate set of production authorization cards for each class. This section deals with the former type of system. Below, we first introduce this model and then describe a method to evaluate the system.
4.5.1 The model

A PAC-based production to order system with $R$ job classes and $N$ universal cards is basically the same as the single-class production to order system with $N$ cards. The main part of the model is the network containing $M$ stations which model the workstations. In addition, there is one synchronization station which comprises one external queue for all jobs and one pallet pool for the universal pallets (see Figure 4.4). However, in the current model, we may identify $R$ independent stochastic processes of external job arrivals. Jobs that arrive when the network is full, wait in the common external queue until they are allowed to enter the network. The jobs leave the external queue in order of their arrival, regardless of the job classes.

The multi-class system with universal production authorization cards differs from the single-class system due to the $R$ different arrival processes mentioned above, and due to characteristics of the network part of the system. As in Section 4.3, we assume that jobs in each class arrive according to a Poisson process, that the workstations and the load-unload station are modeled by stations with general service times and FCFS service discipline, and that the routing for each class is Markovian. However, in the systems that we consider in this section, we allow for class-dependent service times per station, and a different routing mechanism for each class. To be more precise, we consider a model in which class-$r$ jobs arrive according to a Poisson process with rate $\lambda^{(r)}$; class-$r$ jobs circulate inside the network according to a Markovian routing characterized by their routing matrix $P^{(r)}$; and the first two moments of the service times at station $j$ for class-$r$ jobs are given by $E(S_{j}^{(r)})$ and $E(S_{j}^{(r)})^2$.

We introduce some additional notation. This notation concerns variables and vectors identifying the number of jobs (per class) in the various elements of the SOQN-EQ.

$$n_{j}^{(r)} = \text{the number of class-}r\text{ jobs at station } j,$$

$$\vec{n}_{I}^{(r)} = (n_{1}^{(r)}, \ldots, n_{M}^{(r)}) = \text{the class-}r\text{ population vector of the network},$$

$$n_{I}^{(r)} = |\vec{n}_{I}^{(r)}| = \sum_{j=1}^{M} n_{j}^{(r)} = \text{the number of class-}r\text{ jobs in the network},$$

$$\vec{n}_{I} = (n_{1}^{(1)}, \ldots, n_{I}^{(R)}) = \text{the population sizes of jobs in the network},$$

$$n_{I} = |\vec{n}_{I}| = \sum_{r=1}^{R} n_{I}^{(r)} = \text{the total number of jobs in the network},$$

$$n_{E} = \text{the total number of jobs in the external queue}.$$

The universal production authorization cards impose a population constraint on the system. They regulate that the total number of jobs inside the network can never exceed the number of cards:

$$n_{I} \leq N.$$

We assume that the system is stable. That is, we assume that there is both sufficient workstation capacity and a sufficient number of cards to process the offered work. This implies that first of all the service times and the arrival rates
are assumed to be such that the workstations have sufficient capacity to process all arriving jobs, if there would be no PAC constraint:

\[
\sum_{r=1}^{R} \frac{\lambda_{j}^{(r)} \mathbb{E}S_{j}^{(r)}}{c_j} = \sum_{r=1}^{R} \rho_{j}^{(r)} = \rho_{j} < 1, \quad j = 1, ..., M, \quad (4.26)
\]

where \( \lambda_{j}^{(r)} \) denotes the class-\( r \) arrival rate at station \( j \), \( \rho_{j}^{(r)} = \lambda_{j}^{(r)} \mathbb{E}S_{j}^{(r)}/c_j \) denotes the utilization of station \( j \) due to class-\( r \) jobs, and \( \rho_{j} \) denotes the total utilization of station \( j \). The arrival rates \( \lambda_{j}^{(r)} \) follow directly from the routing matrices \( P^{(r)} \):

\[
\lambda_{j}^{(r)} = \lambda^{(r)} P_{0j}^{(r)} + \sum_{i=1}^{M} \lambda_{i}^{(r)} P_{ij}^{(r)}, \quad j = 1, ..., M \text{ and } r = 1, ..., R.
\]

The class-\( r \) visit ratios can be obtained from

\[
V_{j}^{(r)} = \frac{\lambda_{j}^{(r)}}{\lambda^{(r)}}, \quad j = 1, ..., M \text{ and } r = 1, ... R.
\]

From this model, we want to obtain (approximate) typical performance measures such as the mean time a class-\( r \) job spends in the system, the mean number of class-\( r \) jobs in the external queue and the mean number of class-\( r \) jobs in the network.

In [15] Buitenhek presents four approximation methods to analyze multi-class production to order systems with universal cards. Here, we only introduce the complete reduction method, which we also have seen previously in Section 2.4.4 where it was applied to open multi-class manufacturing systems.

### 4.5.2 Complete reduction

First let us recall the three basic steps of the complete reduction method:

1. Reduction of the given \( R \)-class production model into a single-class model by aggregating the \( R \) classes.

2. Analysis of the single-class model.

3. Disaggregation to obtain the performance measures per class for the given \( R \)-class production model.

Step 1 is carried out by adding up the external arrival rates, and by weighing the first two moments of the service times and the routing probabilities with the arrival rates at the stations. Denote by \( \lambda \) the arrival rate, by \( P \) the routing matrix, and by \( \mathbb{E}S_{j} \) the service time at station \( j \) in this single-class model. We
have
\[ \lambda = \sum_{r=1}^{R} \lambda^{(r)}; \quad \lambda_i = \sum_{r=1}^{R} \lambda_i^{(r)}, \quad i = 1, \ldots, M, \]
\[ P_{ij} = \sum_{r=1}^{R} \frac{\lambda_i^{(r)}}{\lambda_i} p^{(r)}_{ij}, \quad i, j = 0, \ldots, M, \]
\[ \mathbb{E}S_j = \sum_{r=1}^{R} \frac{\lambda_j^{(r)}}{\lambda_j} \mathbb{E}S_j^{(r)}, \quad j = 1, \ldots, M, \]
\[ \mathbb{E}S_j^2 = \sum_{r=1}^{R} \frac{\lambda_j^{(r)}}{\lambda_j} \mathbb{E}(S_j^{(r)})^2, \quad j = 1, \ldots, M. \]

The analysis of the single-class production model (Step 2) goes along the lines of Algorithm 7. We give some additional explanation. The visit ratios \( V_j \) in the complement network of the synchronization station in the single-class model must be normalized such that \( V_{M+1} = 1 \) (where \( M + 1 \) now denotes the short, or equivalently, the synchronization station with service time zero). Hence, we obtain
\[ V_j = \sum_{r=1}^{R} \frac{\lambda_j^{(r)}}{\lambda} V_j^{(r)} = \frac{\lambda_j}{\lambda} \sum_{r=1}^{R} \frac{\lambda^{(r)}}{\lambda} \lambda_j^{(r)} = \sum_{r=1}^{R} \frac{\lambda_j^{(r)}}{\lambda} = \frac{\lambda_j}{\lambda}, \quad j = 1, \ldots, M. \]

Algorithm 7 yields the mean total number of jobs in the external queue \( \mathbb{E}L_E \), the mean total number of jobs at station \( j \), \( \mathbb{E}L_j \), and the utilization of station \( j \), \( \rho_j = \lambda_j \mathbb{E}S_j (= TH_j(N) \mathbb{E}S_j) \) for \( j = 1, \ldots, M \). Thus, the performance measures of the single-class model have been obtained.

Equations (2.41) through (2.44) can be used to disaggregate these measures to obtain those per class (Step 3).

4.6 Multi-Class Networks: dedicated cards

The main subject of this section is the analysis of PAC-based production to order systems that produce multiple job classes, while each job class has its own work load restriction. Below, we first describe the model and then introduce a method to evaluate the system.

4.6.1 The model

The multi-class production system with dedicated PAC’s is modeled as follows (see Figure 4.7). The main part of the system consists of the network with \( M \) workstations. In addition to the network, there are now \( R \) synchronization stations, each consisting of one external queue and one card pool. Each synchronization station is dedicated to one class. For class \( r \), there are \( N^{(r)} \) dedicated cards available. The synchronization stations function identically to those described in the previous sections: Class-\( r \) jobs arrive at their external queue and may enter the network if there is a class-\( r \) card in the class-\( r \) card pool. As soon as all operations on the job are completed, the job leaves the system and the
4.6. MULTI-CLASS NETWORKS: DEDICATED CARDS

Figure 4.7: A multi-class PAC-based production to order system with dedicated cards

card is returned to the class-$r$ card pool, where it is available for use by another class-$r$ job.

As in the multi-class system with general purpose pallets, we assume that:
(i) class-$r$ jobs arrive according to a Poisson process with rate $\lambda^{(r)}$; (ii) the service times of class-$r$ jobs at station $j$ have a general distribution with first two moments $E S_j^{(r)}$ and $E (S_j^{(r)})^2$ (service times may be class-dependent); (iii) the service discipline is FCFS at each station; and (iv) class-$r$ jobs circulate inside the network according to the Markovian routing matrix $P^{(r)}$.

In this section, we borrow the notation as introduced in the previous section. Concerning the state descriptions, we have $n_j^{(r)}$, $\tilde{n}_J^{(r)}$, $n_I^{(r)}$, $\tilde{n}_I$, and $n_I$. In addition, we define

$$n_E^{(r)} = \text{the number of jobs in the external queue of class } r,$$

$$\tilde{n}_E = (n_E^{(1)}, \ldots, n_E^{(R)}) = \text{population vector for the external queues.}$$

The dedicated cards impose $R$ population constraints on the network, stating that the number of class-$r$ jobs inside the network may never be larger than the number of class-$r$ cards:

$$n_I^{(r)} \leq N^{(r)}, \quad r = 1, \ldots, R. \quad (4.27)$$

For now we assume that the model is stable. That is, Equation (4.26) holds and thus the workstation capacity is sufficient. (The variables $P^{(r)}$, $\lambda_j^{(r)}$, and $V_j^{(r)}$ are defined as in Section ??.) Furthermore, for each class there is a sufficient number of PAC’s to process the offered work (none of the external queue lengths tends to infinity). The latter conditions are difficult to verify a priori. We come back to this issue at the end of this section, when we study the analysis of unstable systems.

The main goal of the analysis of the multi-class production to order system with dedicated cards is to obtain (estimations of) typical performance measures
such as the mean number of jobs in each external queue, and the mean time in
the system per class.

4.6.2 The extended PDP method

Perros, Dallery and Pujolle [63] propose a generalization of the method of
Dallery [26] (a different method for evaluating single class PAC-based systems)
for the multi-class production to order network with dedicated cards, in which
they assume that a product form solution exists (meaning that under a first
come first serve discipline the service times at each station are exponential and
class-independent). In this section, we generalize this method for networks with
stations that have a general, class-dependent service time. For this we apply
an appropriate AMVA algorithm. We refer to the generalized method as the
extended PDP method. We describe this method as a generalization of the
aggregation method based on the CQN-view, as described in Section ??.

The Basic Idea

The extended PDP method follows the lines of Algorithm 7. So, the first step
is to obtain a CQN-view of the production system with dedicated cards. This
CQN-view corresponds with a multi-class CQN with \( M \) general service stations
and with \( R \) synchronization stations at which the cards are delayed because they
need to consume an external resource (a job). Let the synchronization stations
be numbered such that station \( M + r \) is the class-\( r \) synchronization station.

Once the CQN-view is obtained, the key observation is that \( \text{each synchronization station is only visited by one class} \). In particular, station \( M + r \) is
visited by class-\( r \) jobs and cards only. The basic idea of the extended PDP
method is to replace each synchronization station by a load-dependent (single-
class) server. The load on which the service rates for station \( M + r \) depend is
given by the number of class-\( r \) cards present at the station, and thus also in the
network. This yields a CQN with \( M \) general service stations (the workstations)
and with \( R \) load-dependent (single-class) servers. We refer to this network as
the CQN-LD.

Next, the CQN-LD is analyzed with an appropriate AMVA algorithm to
obtain the performance measures of stations 1 to \( M \), and the performance mea-
sures concerning the cards at the synchronization station (in [63], the Convolution Algorithm is applied). To obtain the performance measures for the external
queues, each synchronization station is analyzed in isolation along the same lines
as explained in Section 4.3 for the single-class production to order system. In
the following subsections we will describe these steps in more detail.

Obtaining the service rates and analysis of the CQN-LD

Once the CQN-view is obtained, the first step is to replace the synchroniza-
tion stations by load-dependent exponential servers to obtain the CQN-LD. As
before, we like to have service rates that result in a consistent analysis. That
is, we require the service rates to be such that the throughput of the cards at
station \( M + r \) is equal to the class-\( r \) arrival rate. It can be verified along the
lines of the analysis presented in Section 4.3 that this is achieved if we set, for
4.6. MULTI-CLASS NETWORKS: DEDICATED CARDS

\( r = 1, \ldots, R, \)

\[
\begin{align*}
\mu_{M+r}(n) &= \lambda^{(r)}, & n &= 2, \ldots, N^{(r)}, \\
\mu_{M+r}(1) &= \frac{\lambda^{(r)}}{q_{M+r}}, \\
\text{with } q_{M+r} &= 1 - \frac{\lambda^{(r)}}{TH_{CM+r}^C(\bar{N})}.
\end{align*}
\]

(4.28)\hspace{1cm} (4.29)\hspace{1cm} (4.30)

However, this replacement is not trivial, because we do not know the value of \( TH_{CM+r}^C(\bar{N}) \). This value depends on all other synchronization stations, in particular on the service rates \( \mu_{M+s}(1) \) for \( s \neq r \). Hence, Equations (4.29) and (4.30), for \( r = 1, \ldots, R \), define a fixed-point problem. This fixed-point problem can be solved with an iterative procedure formulated in Algorithm 8.

Algorithm 8 Iteration to obtain the service rates \( \mu_{M+r}(1) \)

1. (Initialization) Set the service rates for stations \( M + 1 \) to \( M + R \) to some initial value. For example, set \( \mu_{M+r}(1) = \lambda^{(r)} \).

2. (Iteration) Repeat the following steps until convergence of the \( \mu_{M+r}(1), r = 1, \ldots, R \)

   (a) For \( r = 1, \ldots, R \), solve the complement \( C_{M+r} \) of synchronization station \( M + r \) with an appropriate AMVA algorithm. This yields the throughputs \( TH_{CM+r}^C(\bar{N} - n_P^{(r)} \cdot e_r) \) of \( C_{M+r} \), for \( n_P^{(r)} = 0, 1, \ldots, N^{(r)} \), for all \( r \), where \( e_r \) denotes an \( R \)-dimensional vector with its \( r \)-th element equal to one and all other elements equal to zero.

   (b) For \( r = 1, \ldots, R \), reset the service rate \( \mu_{M+r}(1) \) according to Equations (4.29) and (4.30).

The idea behind the calculation of the throughputs of the complement networks or the synchronization stations is that we again assume that Norton’s theorem applies. The appropriate AMVA mentioned in Step 2a of the algorithm can be obtained by extending Algorithm 6 with equations (4.18) through (4.21) for all synchronization stations. Once the service rates are obtained, the CQN-LD can be analyzed with the same algorithm.

Some remarks about Algorithm 8 are in place. First, we cannot mathematically prove that the algorithm always converges. However, in all numerical experiments, not one example was found for which the algorithm did not converge. Second, the initialization of the service rates \( \mu_{M+r}(1) \) does not have to be according to the suggested \( \mu_{M+r}(1) = \lambda^{(r)} \). Another sensible initialization could be \( 1/\mu_{M+r}(1) = 0 \). This choice might even lead to a monotonic convergence of the service rates. Third, in Step 2a, when solving the complement of station \( M + r \), we may use the value of \( \mu_{M+s}(1) \) for \( s < r \) of the previous iteration or the one obtained in this iteration. The latter option may somewhat reduce the number of iterations needed to reach the convergence.
Analysis of the synchronization stations in isolation

Each synchronization station is visited by one class of cards and jobs. The behavior of the station can therefore be analyzed by a birth-death process identical to that in the single-class case. Denote by \( n^{(r)}_P \) the number of production authorization cards at station \( M + r \) and by \( n^{(r)}_J \) the number of jobs at station \( M + r \). We obtain the performance measures for the external queue from the birth-death process with the states \((n^{(r)}_P, n^{(r)}_J)\). This process is obtained from that for the single-class case by replacing \((n_P, n_J)\) by \((n^{(r)}_P, n^{(r)}_J)\); the birth rate \( \lambda \) by \( \lambda^{(r)} \); and the death rates \( \lambda_{M+1}(n_P) \) by \( \lambda_{M+r}(n^{(r)}_P) = TH^{C_{M+r}}(\overline{m} - n^{(r)}_P, \overline{e}^{(r)}) \). These throughput values \( TH^{C_{M+r}}(\overline{m} - n^{(r)}_P, \overline{e}^{(r)}) \) for \( n^{(r)}_P = 0, ..., N^{(r)} \) are obtained already in Step 2a of Algorithm 8. For each job class, this analysis yields the mean number of jobs in the external queue \( E_L^{(r)} \) and thus the expected waiting time.

The algorithm

The extended PDP method is composed of the transformation of the PAC based production system into its CQN-view; the subsequent transformation into, and the analysis of, the CQN-LD; and of the analysis of all synchronization stations. This is formulated in Algorithm 9.

Algorithm 9 The extended PDP method

1. Obtain the CQN-view of the SOQN-EQ.
2. Perform Algorithm 8 in order to obtain the CQN-LD.
3. Analyze the CQN-LD to obtain the performance measures concerning the network and the cards at the synchronization station, \( r = 1, ..., R \), and \( j = 1, ..., M + R \).
4. Analyze station \( M + r \) in isolation to obtain \( E_L^{(r)}_E \), \( r = 1, ..., R \).

Unstable classes and the MQN-EQ

We like to extend the analysis of the multi-class PAC-based production system for unstable systems. In particular, we still assume that each station has sufficient capacity (that is, Equation (4.26) still holds), but we allow that for some classes the number of cards may be insufficient to process the offered work for those classes. To specify the notion of an unstable system, we introduce the notion of unstable classes.

The analysis of synchronization station \( M + r \) in Step 4 of Algorithm 9 can only be performed if we find \( \lambda^{(r)} < TH^{C_{M+r}}(\overline{m}) \) (in fact, some problems would also have been encountered in the determination of \( q_{M+r} \) in Step 2, as we point out later on), otherwise, the resulting birth-death process is not stable. Intuitively, this corresponds to the fact that the external queue length of class \( r \) tends to infinity or, in other words, that class \( r \) is unstable. We deal with unstable classes in two ways. First, we allow classes to be unstable in the model.
and then modify the analysis to account for this. Second, we consider a model in which we deal beforehand with the fact that a class is unstable. The latter case obviously requires that we know beforehand that a class is unstable. In the case of the former, we may be ignorant and only during the analysis phase find out that a class is unstable. Now, let us properly define what we mean by stability and let us describe how to deal with this phenomenon. Denote by \( L^{(r)}(t) \) the counting process that counts the number of jobs in the external queue of class \( r \) at time \( t \).

**Definition 4.4** (Stability and instability) For the PAC-based production to order system:

1. Class \( r \) \((r = 1, ..., R)\) is called **stable** if \( \lim_{t \to \infty} 1/t \int_0^t L^{(r)}(u) \, du < \infty \). The system is called **stable** if all classes \( r = 1, ..., R \) are stable.
2. Class \( r \) \((r = 1, ..., R)\) is called **unstable** if it is not stable. The system is called **unstable** if at least one of its classes is unstable.

Define the class-\( r \) throughput as the number of jobs that leave the production system per time unit, and denote this quantity by \( TH^{(r)}(\bar{N}) \). Then we can formulate the following result.

**Lemma 4.5** For \( r = 1, ..., R \),

1. If class \( r \) is stable, then \( TH^{(r)}(\bar{N}) = \lambda^{(r)} \).
2. If class \( r \) is unstable, then \( TH^{(r)}(\bar{N}) \leq \lambda^{(r)} \).
3. If \( TH^{(r)}(\bar{N}) < \lambda^{(r)} \), then class \( r \) is unstable.

We have already used part 1 of Lemma 4.5 in the design of Algorithm 9: the load-dependent service rates of each synchronization station are chosen such that the throughput of the jobs equals the arrival rate of the jobs. However, according to part 2 of Lemma 4.5, for unstable classes, we may find a throughput that is smaller than the arrival rate and therefore, we have to redesign our solution method. Below we describe how to deal with this.

Suppose we are given a multi-class PAC-based production to order system and we are ignorant of the fact that there are unstable classes. Let us say, however, that class \( r \) is unstable. When solving the fixed-point problem (identifying the load-dependent service rates), it may then occur in some iteration that the \( TH^{C_{M+r}}(\bar{N}) \) is smaller than \( \lambda^{(r)} \). If we would continue our analysis as before this would lead to \( q_{M+r} < 0 \). In other words, the service time for a card at station \( M + r \) may be negative. However, the analysis should not have been continued, because the result for \( q_{M+r} \) is only valid if the birth-death process is stable. Therefore, we need a different solution for this case. We resolve this situation as follows. If we find \( TH^{C_{M+r}}(\bar{N}) < \lambda^{(r)} \), we reset \( q_{M+r} = 0 \), which corresponds to setting \( \mu_{M+r}(1) \) to infinity in the current iteration. In other words, we set the service time of this class at its synchronization station to zero **in the current iteration**.

Notice that the value of \( q_{M+r} \) obtained in an iteration of the solution procedure does not have a useful interpretation for the given model. It is just part of
the solution examined in that iteration. The value of $q_{M+r}$ in the next iteration may well satisfy $0 < q_{M+r} < 1$. We only conclude that a class is unstable in the given multi-class model if after convergence of step 2, we still have $q_{M+r} = 0$, and therefore after convergence of the procedure and analysis of the resulted CQN-LD we find that $TH^{(r)}(\bar{N}) < \lambda^{(r)}$.

Now suppose we are given a multi-class production model and we know that class $r$ is unstable. Then we also know that, upon the arrival of a class-$r$ card at the synchronization station, there is always a class-$r$ job in the external queue. Therefore, the card and the first job in its external queue immediately synchronize and together enter the network. In other words, the service time of class-$r$ cards at their synchronization station is always zero. The class-$r$ throughput $TH^{(r)}(\bar{N})$ in the production network is equal to the throughput of the cards as measured at the card feedback loop of the network which arises after removing the synchronization station and only considering the cards of this type. In the sequel, classes without synchronization stations are called \textit{closed classes}. On the other hand, classes for which the service time may be positive at the synchronization station are called \textit{open classes}. We refer to networks with open and closed classes and pallet constraints as \textit{Mixed PAC-based Production Networks.}

The analysis of the Mixed PAC-based production network may be desirable because one wishes to obtain the maximum throughput for one specific part type. This value is obtained by first constructing PAC-based production model of the system to be studied, then closing this part type, and finally analyzing the Mixed Network thus obtained.

If all part types in an Mixed PAC-based production network are closed, we have the traditional Closed Queueing Network model. On the other hand, if all types are open, we have the PAC-based production to order system. Thus, both the latter networks are special cases of the Mixed PAC-based networks.

The analysis of Mixed PAC-based production network goes along the same lines as the analysis of the PAC-based production to order model. In particular, the Mixed Network can be transformed into an equivalent CQN-view in which all classes are closed (Step 1). The open classes in the Mixed Network have then become closed classes with a synchronization station and the closed classes are unchanged and thus do not visit any synchronization station (the real closed classes). In Step 2, the CQN-view can be transformed into a CQN-LD by replacing the synchronization stations by load-dependent (single-class) exponential servers. The analysis of the CQN-LD is performed with an AMVA algorithm which has to deal with $M$ general service stations and $R_O$ load-dependent servers, where $R_O$ denotes the number of open classes.

In the mixed networks, open classes may be unstable. If we know this beforehand, we can of course close these classes to obtain a different mixed network with more closed classes. Otherwise, the modified algorithm for this case deals with the unstable classes in the same way as with the unstable classes in the PAC-based production to order system.
Bibliography


