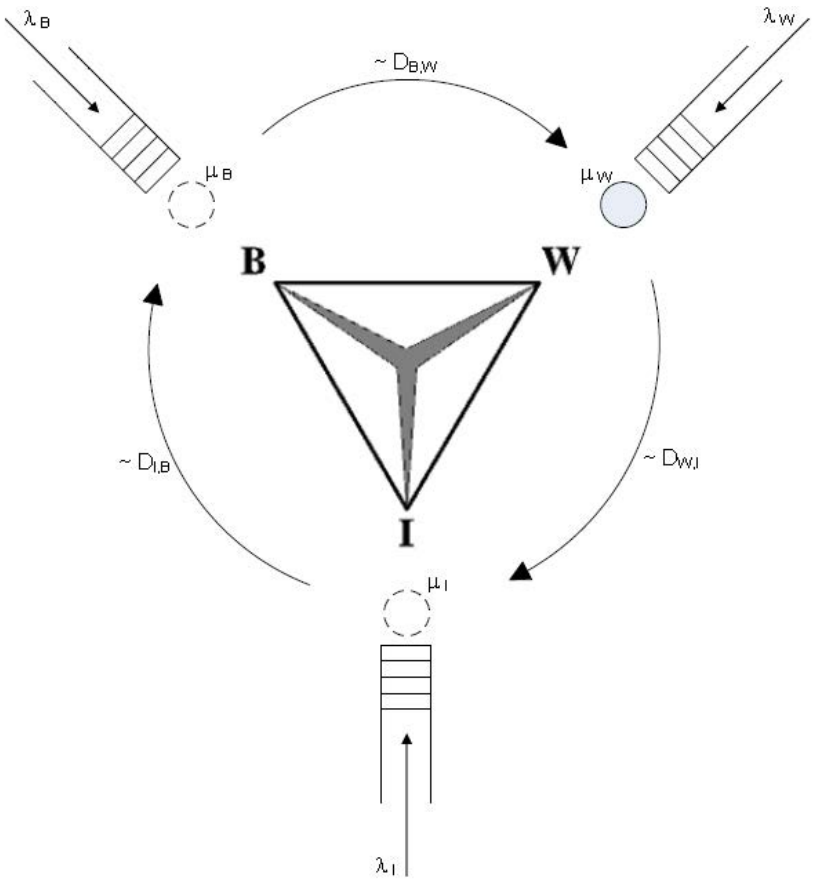
BMI Paper

Variance effects in polling systems

The limits of Japanese production theory?



Author:
Jan-Pieter Dorsman

Supervisor:
Dr. Ir. Erik Winands

BMI Paper

Variance effects in polling systems

The limits of Japanese production theory?

Author:
Jan-Pieter Dorsman

Supervisor:
Dr. Ir. Erik Winands

August 2009

Vrije Universiteit Amsterdam
Faculty of Sciences
De Boelelaan 1081a
1081 HV Amsterdam

Preface

This paper was written as part of the *Business Mathematics and Informatics* master programme. The goal of the BMI paper is to give an answer to a certain problem statement by means of research. The paper should encompass at least two of the following programme components: 'business', 'mathematics' and 'informatics'.

In this paper we consider polling systems. More specifically, we study the effect of the system policies on the behaviour of the queue lengths, when set-up times are cut. This is done mainly by means of simulation. The simulation part accounts for the mathematics and informatics components of the programme, while the business part is represented by polling systems themselves; polling systems are for example widely used in production facilities.

I would like to thank my supervisor dr. ir. E.M.M. Winands for introducing me to the subject of polling systems and for his support and feedback during the process of writing this paper.

Abstract

In this paper we consider polling systems, systems in which one single server serves jobs from multiple queues. This server typically needs a set-up time to switch from one queue to another. More specifically, we study the effect of cutting these set-up times on the lengths of the queues. In case of the so-called exhaustive service policy and the cyclical sequencing policy, it is shown in literature that the lengths of each of the queues in the system can in fact grow without bound as the duration of the set-up periods is being cut. As a result, a discussion is ongoing whether Japanese production theory, which prescribes that set-up time should be limited, is flawed. After all, a decrease in set-up time does not guarantee an improvement on the queue lengths. An argument against this statement is that the policies used in the example are sub-optimal and hence the statement is not valid. According to this argument, the apparent sub-optimality of the policies is to blame rather than the Japanese production theory.

Our main research objective is to find out if this effect exclusively occurs when these policies are used or that a wider group of policies can provoke this effect. For this purpose, we consider several policies common in practice. Because analysis is not always tractable for these policies, we study the behaviour of the queue lengths using these policies by means of discrete time event simulation.

We observe that the queue lengths grow without bound when set-up times are being cut, in case the policies used do not allow the server to idle and force the server to always either be serving a job or setting up for service at another queue.
When the policies used allow the server to idle at queues and be in stand-by mode, the queue lengths no longer grow without bound and the variance effect loses a lot of its strength.
Since policies not allowing the server to idle are used commonly in practice, we conclude that the claim of the existence of flaws in the Japanese production theory should not be dismissed based upon arguments attacking the representativity of the policies considered in literature.

Contents

Chapter 1

Introduction

Half a century ago, the principle of the Just-In-Time production (JIT) originated in Japan, also referred to as the concept of Japanese production theory. This theory states that inventory in production systems reflects 'waste'. Everything in the production facility that does not add value, but does increase production time and costs can be seen as waste. Inventory is thus regarded as such. As a consequence, this theory claims that the need of inventory reflects flaws in a production system. JIT states that by reducing inventory, one is forced to address these flaws, which in general results in better performance of the production system in terms of production time and production costs. This principle is well celebrated in practice.

In 1991, Sarkar and Zangwill[10] examined a certain type of a production system called a polling system. In polling systems, there is one machine or server processing jobs at multiple queues. When the serves moves from one queue to another, the server needs time to prepare for processing the new type of jobs. This is called set-up time or switch-over time. Following the Japanese production theory, reducing this set-up time should yield a reduction of the inventory, in this case the lengths of the queues.
Sarkar and Zangwill however show that this assumption may be erroneous in certain circumstances! Indeed, they show that queue lengths can go up if set-up time is reduced, which is a very counter-intuitive and anomalous effect.

In a subsequent provocative paper, Zangwill[13] contests the Japanese production theory based on this discovery:

> *"The paradoxes and inconsistencies in Japanese production theory seem to have a pattern and occur in certain situations. If we uncover the limits of that theory, we might discover a new and more advanced theory of production and productivity."*

This paper drew a massive amount of response, the executive editor stated that this article alone drew more response than all other articles combined during his editorship. Following up on this article, Duenyas[2] and Gerchak and Zhang[4] both claim that the observed effect is not a flaw in the reasoning of Japanese production theory as much as it is a flaw in the reasoning of Zangwill. The example given by Zangwill in his provocative paper supposedly uses a system policy[1] that is not nearly optimal and that would be the reason for the observed

[1] A set of rules that specifies what action the server should take, like processing a job, setting up for service at another queue or even be stand-by and do nothing at all, given information about all of the other variables in the system, such as the lengths of the queues.

anomalous effect to occur. Therefore, they claim the Japanese production theory is not in error.
In response to these papers, Zangwill[14] does not deviate from his original view on the matter. His point of defense is that whether something is optimal or not does very much depend on what one would like to optimize. If the objective is to minimize the inventory in the whole system, Zangwill claims that the policy used is at least close to optimal.

In this discussion no strongly founded arguments (or maybe even a proof) can be found that back up the claims that sub-optimality of the policy is or in case of Zangwill is not the key issue. Duenyas states that another, supposedly more optimal policy does not exhibit the described effect. However, there is no guarantee available that this policy is in fact optimal, it might be sub-optimal as well. Gerchak and Zhang express their high doubts about the system policy used in Zangwill's example being optimal, but again it is not shown rigorously that the sub-optimality is the very reason why this effect occurs.
Zangwill in return attacks the optimality of the policy by Duenyas and gives other examples of polling systems exhibiting the anomalous effect in which he tries to take away the doubts of Gerchak and Zhang about the optimality of his own policy. However, the question whether it is sub-optimality of the policy itself that can cause this effect remains unaddressed.

The research objective of this paper is to research whether the policy is a major factor in what triggers the anomalous effect. More specifically, we want to determine whether it is only the specific policy used by Sarkar and Zangwill that shows the effect or that the effect is more generally existent in a wider group of policies. If the latter is the case, we aim to characterize this group of policies. Consequently, we try to conclude whether this effect is merely theoretical or should be considered in practice policy-wise.

Since in case of most policies there is no closed-form mathematical expression known for the length of the separate queues, this research will be done by means of a simulation study. A simulation tool was exclusively made for this paper to study the behaviour of the several polling systems. This tool imitates the behaviour of a polling system with given characteristics. Meanwhile, the tool also acts as an observing outsider, who keeps track of this system at all times, and is therefore able to give an answer on the question how certain things like its queue lengths behave.

In chapter 2, the characteristics of a polling system are described more carefully and the set-up time effect found by Sarkar and Zangwill[10] is also examined more closely. Furthermore, an overview of the policies we will study by means of simulation is presented. In chapter 3, for each policy several polling systems are considered and simulation runs performed. Of course, we run simulations with both high and low set-up times. In addition to giving the results of the simulation, we aim to give an intuitive explanation of these results. Finally, in chapter 4 we aim to fulfill our research objective and give an answer to the questions at hand.

Chapter 2

Overview of policies

2.1 The polling system

A polling system consists of n queues ($n > 1$) at which jobs (products or customers) arrive, waiting to be processed by one single server (a machine or person processing products or serving customers) processing jobs at each of the queues. The server cannot serve all queues simultaneously, only one queue is served at a time. For a schematic view of a polling system with three queues, see Figure 2.1.
At each queue i, jobs of type i arrive according to a certain arrival process, of which the interarrival times are distributed according to a random variable A_i with expectation $\mathbb{E}A_i$ and variance $\mathrm{Var}(A_i)$. Throughout this paper the assumption of exponential interarrival times with a certain rate λ_i is made, in which case the arrival process is a Poisson process with rate λ_i.

When the server is currently set up to serve queue i, it has the possibility to process a job at queue i. This incurs a processing time (also called service time), which again is a random variable denoted by S_i with expectation $\mathbb{E}S_i$ and variance $\mathrm{Var}(S_i)$. The server also has the possibility to stop serving the current queue i at all and set up for or switch to another queue j ($i \neq j$), since products or customers at other queues are waiting to be processed as well. This typically takes set-up time or switch-over time, also called a set-up period. Finally, the server can also decide to idle at queue i, that is stay at the current queue in stand-by mode until some event occurs, after which it decides to commence service or set-up once again.

Whether to stay idle or be active (serving a job or setting up for service at a queue) is a decision made by the *idling policy*. In case the server will be active and becomes available at a queue after just having arrived at the queue, having processed a job at the queue or just having concluded an idling period at the queue, the question whether to process another job at the current queue or to start setting up for another queue is addressed by the *service policy*.

Leaving a queue i and setting up for service at queue j incurs set-up costs k_{ij} and a set-up time d_{ij}, since changes have to be made to the server in order to prepare for the processing of a new type of job at the cost of resources like money and time. The k_{ij}-parameters are assumed to be constant and known beforehand, while the set-up times are realisations from the random variables D_{ij} with expectation $\mathbb{E}D_{ij}$ and variance $\mathrm{Var}(D_{ij})$.

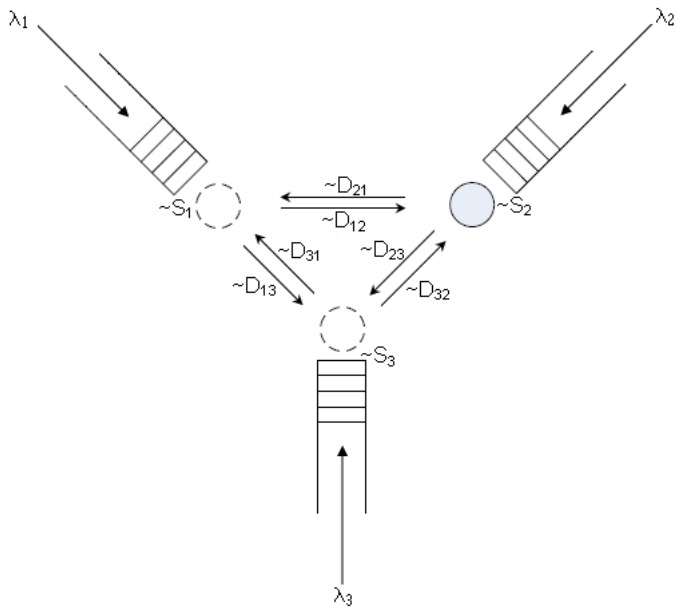


Figure 2.1: *A polling system with three queues.*

When the set-up costs are not specified, they are assumed to be equal for every possible set-up from one queue to any other, in which case they do not need to be considered explicitly when evaluating the performance of the system.
In the literature, it is often assumed that D_{ij} is the same for each possible i, $i \neq j$. When this assumption is made, we use the random variables D_j, because the index i has become redundant.
The decision which queue j to set up for, after having decided to leave queue i is determined by the *sequencing policy.*

W_i is the random variable representing the waiting time in queue i with expectation $\mathbb{E}W_i$. The length of queue i at a certain point in time when the system is stable is represented by L_i, which has expectation $\mathbb{E}L_i$. If the arrival process is a renewal process such as the Poisson process, Little's law should hold: $\mathbb{E}L_i = \lambda_i \mathbb{E}W_i$. From this it follows that an observation of overall increasing (decreasing) waiting time in queue i will always go together with the observation of an overall increasing (decreasing) length of queue i, since we assume exponential interarrival times.
The average waiting time W is defined as $\sum_{i=1}^{n} \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j} W_i$ and has expectation $\mathbb{E}W$. The load of queue i is defined as $\rho_i = \frac{\mathbb{E}S_i}{\mathbb{E}A_i}$. The server utilization ρ is then defined as the sum of these loads: $\rho = \sum_{i=1}^{n} \rho_i$.

2.2 The anomalous effect of reducing set-up times

Sarkar and Zangwill[10] published a paper in 1991 that showed that reduction of set-up times in polling systems can in contradiction with the JIT-principle force the mean queue lengths to go up. This is a very counter-intuitive effect, since one would expect the queue lengths to fall, because by reducing set-up times, the server would have more time to process jobs,

which would be in favour of the queue lengths and the job waiting times.
Consider a polling system with n queues, each with a Poisson arrival process with rate λ_i. The server keeps processing jobs at each queue i till the queue is completely empty and there are no more jobs of type i to process. After that, it immediately starts setting up for queue $i+1$, or in case the server just cleared queue n, it starts setting up for queue 1 again. The server is always active and never idles. Using this so-called exhaustive service policy and cyclic sequencing policy, Sarkar and Zangwill show that the following holds:

$$\mathbb{E}W_i = \frac{1-\rho_i}{2}\Big(\frac{\sum_{j=1}^n \mathbb{E}D_j}{1-\rho} + \sum_{j=1}^n h_{ij}(\rho_1,...,\rho_n)\{\lambda_j \mathbb{E}(S_j^2) + \lambda_j \mathrm{Var}(S_j) + \frac{\mathrm{Var}(D_j)}{\sum_{k=1}^n \mathbb{E}D_k}(1-\rho)\}\Big) \tag{2.1}$$

where the h_{ij} are functions depending only on the ρ_i and yield possible values. Sarkar and Zangwill[9] show that these functions are known explicitly and are derived by solving a linear system of n equations in n variables.

When looking at equation 2.1, we see that the set up times contribute to the mean waiting time of queue i through the following two terms:

$$\frac{1-\rho_i}{2}\Big(\frac{\sum_{j=1}^n \mathbb{E}D_j}{1-\rho}\Big)$$

and

$$\frac{1-\rho_i}{2}\Big(\frac{\sum_{j=1}^n \mathrm{Var}(D_j)}{\sum_{j=1}^n \mathbb{E}D_j}(1-\rho)\sum_{j=1}^n h_{ij}(\rho_1,...,\rho_n)\Big)$$

since the several D_j are independent. When cutting the set up times (that is, decreasing $\sum_{j=1}^n \mathbb{E}D_j$), the situation might occur that $\frac{\sum_{j=1}^n \mathrm{Var}(D_j)}{\sum_{j=1}^n \mathbb{E}D_j}$ grows sufficiently large such that the increasing effect on the second term overshadows the decreasing effect on the first term. In this case, the sum of both terms has increased with respect to the original situation, which results in overall longer waiting times, and as a consequence higher queue lengths. This result goes against intuition.
The key issue here is that when cutting down the sum of expected set-up times or the total expected set-up time incurred during one cycle[1] $\sum_{j=1}^n \mathbb{E}D_j$, the sum of the several set-up time variances $\sum_{j=1}^n \mathrm{Var}(D_j)$ does not necessarily decrease. Moreover, even if $\sum_{j=1}^n \mathrm{Var}(D_j)$ would decrease but not at the same rate as $\sum_{j=1}^n \mathbb{E}D_j$, $\frac{\sum_{j=1}^n \mathrm{Var}(D_j)}{\sum_{j=1}^n \mathbb{E}D_j}$ could still grow without bound and trigger the anomalous effect of higher queue lengths. Consequently, one should not only be concerned about the mean of the set-up times, but also the degree of variation they exhibit!
We will commonly refer to this effect as 'the variance effect' or 'the anomalous effect'.

Example 2.1 *We consider the described situation with two queues in a production system. Suppose that* $\lambda_1 = \lambda_2 = \frac{1}{4}$ *and the processing times are exponentially distributed with* $\mathbb{E}S_1 = \mathbb{E}S_2 = 1$.

[1] The start of service at an arbitrary queue till the next start of service at the same queue.

Initially, the set-up times are characterized as follows: $\mathbb{E}D_1 = \mathbb{E}D_2 = 3$, $Var(D_1) = 0$, $Var(D_2) = 900$. *In this case the time to set up for queue 1 is deterministic. The high variance of the time needed to set-up for queue 2 seems very unnatural, however in cases where the server has a chance to break down and consequently needs a lengthy repair these numbers are realistic.*
Under these circumstances, $\mathbb{E}L_1 = 24.125$ *and* $\mathbb{E}L_2 = 16.625$. *The management of this production systems now wants to address these queue lengths in accordance with the JIT-principle and decides to reconsider the procedure of the server set-up to queue 1. A successful result is achieved; possibilities to cut down* $\mathbb{E}D_1$ *to 1 are discovered. This set-up time remains deterministic. However, upon implementing this change, management in this case found that in contrary to their intentions the queue lengths have risen! Now,* $\mathbb{E}L_1 = 35$ *and* $\mathbb{E}L_2 = 23.75$, *which is an increase of more than 40% for both queues!*

The difficulty level of the analysis of polling systems ranges from easily doable to very hard, depending on the scenario. While analysis of this particular scenario is tractable, the analysis of scenarios with other service, idle or sequence policies might not.

2.3 Service policies

Whenever the server becomes available for service at a queue after a service period, set-up period or even an idling period in which the server was put on stand-by, the question whether the server needs to process jobs at the current queue or set up for service at another queue, given that server will be active, is addressed by the service policy.
We consider the following service policies:

Service policy	Description
Exhaustive policy	The server does never switch to another queue if there are jobs left to be processed in the current queue.
Quantity-limited policy	At each queue i the server will process jobs until either k_i jobs have been processed or until the queue is empty, whichever happens first. Afterwards the server will switch.
Time-limited policy	At each queue i the server will process jobs until either t_i units of time have past or the queue is empty, whichever happens first. If the queue threshold of t_i has been reached, the server will finish processing the current job. Afterwards the server will switch to another queue.

In the following subsections each of these service policies is examined more closely. Of course, in practice variations or combinations of these policies and completely other policies exist.

2.3.1 Exhaustive policy

In situations where the server does not start setting up for another queue while there are jobs waiting in the queue which the server is currently set up for, one speaks of an exhaustive

service policy. In practice this service policy is widely used in cases where the jobs of the different queues have equal importance and thus the only performance measure is the amount of work to be done in the whole system. In case of a production system, it is indeed intuitively plausible that one server should continue processing jobs of the queue it is currently at rather than setting up for the next queue which incurs a set-up time, which in a way can be seen as downtime. When processing another job at the current queue, new jobs can arrive at each queue during the processing time of this job, maximizing the time between two set-up periods. Levy, Moshe and Boxma[5] indeed show that exhaustive-like service policies are dominant to other types of service policies when one is only interested in the minimization of the amount of work, under the assumption that the server never idles.

However, it is not always the case that the amount of work in the system is the only thing that one should look at. Some queues may be more important to others, because jobs waiting to be processed in those queues are of a very desired type or the holding costs of inventory of those queues are much higher.
In systems where the jobs demanding service at queues are customers (customer systems), 'fairness' often plays a role as well. Serving another customer who just arrived at the queue the server is at while customers at other queues are waiting a long time already can yield such a high amount of what is called 'unfairness', that an exhaustive policy is out of question.

When the server is never idling and the sequence policy is cyclic(see section 2.4.1), the conditions that the set up times are finite and that $\rho < 1$ are necessary and sufficient for the queue lengths not to be growing without bound (i.e. for the system to be stable). This is shown by Fricker and Jaibi[3].

In case of the exhaustive service policy, analysis is tangible in cases when the service sequence is cyclic (or according to a pre-defined table) and the server is always active (that is either processing jobs or setting up for service of a current queue), as was seen in section 2.2.

2.3.2 Quantity-limited policy

When the server adheres to a quantity-limited service policy, the server can only process so many jobs at a certain queue before it starts to set up for the next queue. For each queue i a parameter k_i is defined, which denotes the amount of jobs the server is able to process during one service period at this queue. If the queue becomes empty and the server has not processed k_i items yet, the server will end its current service period prematurely and start setting up for the next queue in line.

This service policy is used in practice for several reasons. Sometimes, technical restrictions apply to production of units of a certain type. For example, in a production facility where chemical substances are created, the server might not be able to keep producing for an indefinite amount of time; after having provoked a certain amount of chemical reactions, the ability of processing another batch of chemical substance has faded away and the server needs to set up for the provoking of other chemical reactions.
Another reason why this policy might be favoured is because of the ability to prioritize among different queues. When queue i is more important than queue j, k_i can be set higher than k_j in which case more jobs will be processed at queue i than at queue j during one service

period.
In terms of fairness in customer systems this policy also performs better than for example the exhaustive policy. When the k_i-parameters are set with that in mind, a situation can be achieved such that the waiting time for an arbitrary customer is not heavily dependent on the queue in which the customer arrives.

It occurs that when using this service policy, waiting times in some queues may grow without bound, while the other queues appear to be stable. In cases where the service sequence is cyclic(see section 2.4.1) and the arrival process of each queue i is Poisson with rate λ_i, a necessary and sufficient condition for stability at each queue, and thus the whole system, using this service policy is derived and proven by Fricker and Jaibi[3]. When we define the sum of all set-up times in one cycle as a random variable D with expectation $\mathbb{E}D$, this condition reads:

$$\rho + \mathbb{E}D \max_{i \in \{1,\ldots,n\}} \frac{\lambda_i}{k_i} < 1 \tag{2.2}$$

In case $k_i = \infty$ for each i, this service policy is equivalent to the exhaustive policy. This service policy may be referred to as the k-limited policy if $k_1 = k_2 = ... = k_n = k$.
Analysis of polling systems with this service policy is in general hardly tangible.

2.3.3 Time-limited policy

The time-limited policy is similar to the quantity-limited policy because this policy also limits the amount of work the server can do at a certain queue before switching to another queue. However, in case of the time-limited policy, the duration of the service period is not restricted by a certain job limit, but by a certain time limit. For each queue i, a parameter t_i is defined, which denotes the limit of time the server is able to spend on this queue before a set-up to another queue is needed. If the queue becomes empty and the server has not spent t_i units of time processing jobs of that type, the server will end its current service period prematurely and start setting up for the next queue in line. If the time limit has been reached, the server will switch to the next queue in line after it has ended processing the job which was in process at the moment the time limit expired.

The motivations for using this service policy are quite the same as the quantity-limited policy. A time limit could be imposed by a technical constraint, it can be a means of prioritizing queues, and it could also be used to control fairness in customer systems.

In case $t_i = \infty$ for each i, this service policy is equivalent to the exhaustive policy.
In case of deterministic service times, this service policy is equivalent to the quantity-limited policy. A time limit of t_i units of time then amounts to exactly one job limit k_i through $k_i = \lceil \frac{t_i}{\mathbb{E}S_i} \rceil$.
Analysis of polling models with this service policy is in general hardly tangible.

2.4 Sequencing policies

The sequencing policy decides what queue to switch to, when the server is entering a set-up period. The sequencing policies considered are the following:

Sequencing policy	Description
Pre-set sequencing	The server visits the queues repeatedly in a fixed, predetermined order.
Longest Queue First (LQF)	When initiating set-up, the server will switch to the queue which has the most jobs waiting at that moment.
Prioritize by characteristics	When initiating set-up, the server will switch to the queue which sustains the longest time since service, weighted by a certain queue characteristic, such as the load or arrival rate.

In the next subsections, we will take a closer look at these policies.

2.4.1 Pre-set sequencing

In cases where the server visits the queues in a fixed sequence known beforehand, one speaks of a pre-set sequencing policy.

A special case of this sequencing policy is the cyclical sequencing policy, in which the queue sets up for service at queue $i + 1$ after having processed jobs at queue i. In case the server just stopped service at queue n, the server will set up for queue 1 again. The start of service at queue 1 till the next start of service at queue 1 is called a cycle.

Pre-set sequencing policies are commonly used in polling systems. One common reason for this is the existence of technical constraints, such that from one queue the server cannot set up directly to any other queue. Consider for example a production system in which paint is produced. After the server has produced black paint, it cannot go straight to producing white paint, because there are still traces of black paint left in the machine. Instead, the machine needs to produce other colours of paint first, like several darker and lighter shades of grey.
Also, in systems where information about the length of each queue is not completely available, the cyclic policy is a good sequencing policy to adhere to. Indeed, Liu and Nain[6] show that in cases where only the history of each queue up till the last time the server visited that queue is known and an exhaustive service policy is used, a cyclic policy should be adopted when one is only interested in minimizing the total amount of work in the system.

2.4.2 Longest Queue First (LQF)

When the server has a total freedom of set-up[2], a self-evident sequencing policy is the Longest-Queue-First policy. When using this policy, the server will set up to the queue containing the highest amount of jobs, after having stopped service at the current queue.
The idea behind this sequencing policy is simple; in general long queues are undesirable, so in order to contest these, the server switches to the longest queue. Also, in case of an exhaustive policy, the service periods will typically be the longest using this sequencing policy. Over time less set-ups will be needed, which in a way accounts for less downtime.

[2]The server is at each queue able to set up to any other queue.

Liu and Nain[6] show that in case an exhaustive service policy is used and information about the length of all queues is known continuously, one should adopt the Longest-Queue-First policy in order to minimize the total amount of jobs in the system. However, this policy is based on minimizing the total amount of work in the system. When one aims to differentiate between queues because one is more important than others, this sequencing policy will not always perform well.

2.4.3 Prioritize by characteristics

When information about the queue lengths is not readily available at all times, however information about the characteristics of each queue is available (distribution interarrival time, distribution service time, load, waiting space in the queue, time since last visit), the management may base decisions which queues to switch to on these characteristics.

Let l_j denote the time since the latest service period at queue j and suppose the server currently is at queue i. We list several decisions, which will also be examined in the simulation study:

1. Let $k = \text{argmax}_{j\in\{1,..n\},j\neq i}\, l_j\rho_j$. After stopping service at queue i initiate a set-up to queue k. Here, queues with high loads are favoured. We call this policy the PHVL sequencing policy (Prioritize High Valued Load).

2. Let $k = \text{argmax}_{j\in\{1,..n\},j\neq i}\, \frac{l_j}{\mathbb{E}A_j}$. After stopping service at queue i initiate a set-up to queue k. Here, queues with low interarrival times are favoured. We call this policy the PLEA sequencing policy (Prioritize Low Expected interArrival time).

3. Let $k = \text{argmax}_{j\in\{1,..n\},j\neq i}\, l_j\mathbb{E}S_j$. After stopping service at queue i initiate a set-up to queue k. Here, queues with high service times are favoured. We call this policy the PHES sequencing policy (Prioritize High Expected Service times).

4. Let $k = \text{argmax}_{j\in\{1,..n\},j\neq i}\, \frac{l_j}{\mathbb{E}S_j}$. After stopping service at queue i initiate a set-up to queue k. Here, queues with low service times are favoured. We call this policy the PLES sequencing policy (Prioritize Low Expected Service times).

These policies each prioritize by a certain characteristic: the queue load, the interarrival time and the service time respectively. They are then weighted by the time since the last visit, so that lesser favoured queues will be visited as well after time elapses. They may be used in practice because of their self-evidence; management might feel for example that queues with a high load yield more urgency.

2.5 Idling policies

Whenever the server ends processing a job or just concluded a set-up period, the server has the possibility to either be active[3] or to idle at a queue. A server is said to be idling if it is in stand-by mode, ready to resume activity instantly. The question whether to idle or not is addressed by the idling policy. The following idling policies are considered:

[3]Processing jobs or setting up for service at another queue.

Idling policy	Description
No-idling	The server does never idle, it is always either processing a job at a certain queue or in a set-up procedure.
Empty-system-idling	Whenever each queue has no jobs waiting to be processed, the server goes in stand-by mode at the queue it processed its last job at. The server will go back into active mode (service or set-up), once a job arrives at any queue.
Switch to important queue first	Whenever the system is empty, the server will switch to the queue that is deemed most important based on a certain characteristic, after which the server idles at that queue till the next arrival to the system.

In the next subsections we will examine these idling policies more closely. Again, in practice other idling policies do of course exist.

2.5.1 No-idling

When using the no-idling policy, the server is always either processing jobs or setting up at any time in all possible circumstances. In short, the server is never idling.
In some situations, the option to do this may be optimal. Although it may intuitively seem optimal for the server to be stand-by and idling at some queue instead of being totally irresponsive while in the middle of a set-up period, idling does not always yield better results, as we will see in example 2.2.

Example 2.2 *Consider an empty system with two queues, where the server just cleared queue* 1 *of its jobs. We assume that both queues have equal characteristics when it comes to arrival processes, service processes, et cetera. Now, it needs to make the decision whether to idle at queue* 1 *till the next arrival of a job in the system or to switch to queue* 2*, incurring a set-up time of two time units. We consider the decisions of idling and no idling in two different scenarios. These are depicted in Figure 2.2.*

Assume that in the first scenario the system is currently empty, the next arrival at queue 1 *occurs after one unit of time and the next arrival at queue* 2 *will not be occurring until much later. In this case the decision to idle would have been better, because the server could have immediately started processing the job upon arrival, whereas in the other case it would have been right in the middle of a set-up period to the wrong queue and the job would have needed to wait in queue* 1 *for three units of time.*

However, another plausible scenario is that the first arrival to the system is at queue 2 *after one unit of time and there is no arrival at queue* 1 *until much later. In this case, the decision of switching immediately would have been in favour. When the server idles till the arrival at queue* 2*, the server realizes it is at the wrong queue, and hence the job at queue* 2 *will need to wait for a whole set-up period to pass by. In the no-idling case however, the set-up period would have been partly completed already at the time the job arrived at queue* 2*.*

In cases where analysis of a polling model is tangible, usually the server is never idling.

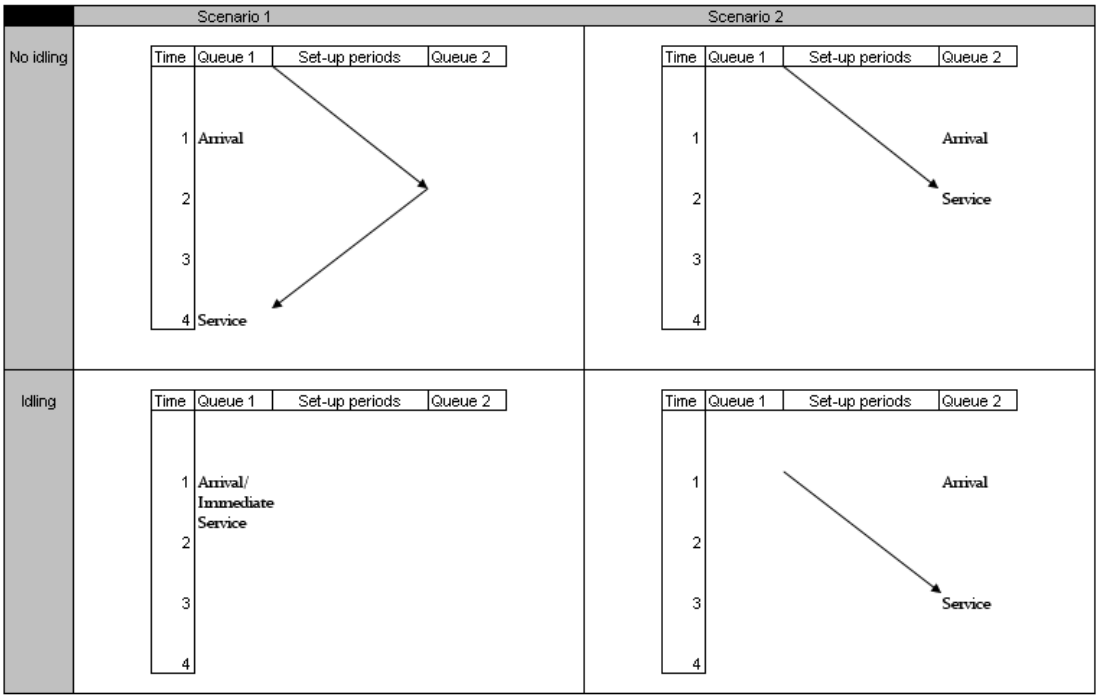


Figure 2.2: *Idling decisions in two different scenarios.*

2.5.2 Empty-system-idling

When the system is empty, the server can opt to go in stand-by and idle at the queue at which jobs of the latest processed job type arrive.

As was shown in example 2.2, this strategy can have benefits in certain situations, and can have a decreasing effect on the total amount of jobs in the system. Moreover, Liu and Nain[6] show that in a fully symmetric polling system, idling at the last visited queue is optimal, whenever the system is empty.

2.5.3 Switch to important queue first

The idea behind the 'Switch to important queue first'-idling policy is that in case of an empty system the server will first switch to the queue deemed to be most attention deserving because of some characteristic of that queue. Then, the server will idle there if there are no arrivals to the system in the meantime. The benefit of this idling policy is that the server will be available at the queue of choice, such as the one with the highest arrival rate, rather than just an arbitrary queue which was last visited.
There are several possible variations of this policy. The following variations are examined in our simulation study:

1. Let queue j be the queue with the highest load. When the server is currently at queue i, switch to queue j if $i \neq j$ and idle there. Idle at queue i otherwise. We call this policy the HL idling policy (Highest Load).

2. Let queue j be the queue with the lowest interarrival time expectation. When the server is currently at queue i, switch to queue j if $i \neq j$ and idle there. Idle at queue i otherwise. We call this policy the LIA idling policy (Lowest InterArrival time expectation).

3. Let queue j be the queue with the highest service time expectation. When the server is currently at queue i, switch to queue j if $i \neq j$ and idle there. Idle at queue i otherwise. We call this policy the HES idling policy (Highest Expected Service time).

4. Let queue j be the queue with the lowest service time expectation. When the server is currently at queue i, switch to queue j if $i \neq j$ and idle there. Idle at queue i otherwise. We call this policy the LES idling policy (Lowest Expected Service time).

This idling policy is used in cases where the mean switch-over times are typically low compared to the arrival rates of the queues. In those cases it is likely that there are no arrivals during the switch-over period, such that there is no effective downtime.

2.6 Heuristic by Van Oyen and Duenyas

While going through the several service, idling and sequencing policies in the previous sections, we have mainly looked at policies that are commonly used in cases where the waiting of jobs of each type is equally undesirable. However, in practice sometimes it is desired that the size of inventory at one queue is pushed back at the cost of having more inventory at another queue. Inventory costs could be much higher at one queue than another or the management might feel the need to prioritize the production of a certain type of job. In those cases, holding costs parameters c_i are introduced for each queue, which denote the cost of having one job of type i waiting for one unit of time at queue i. Of course, in a polling system in which the server is free to set up from each queue to every other, the server tends to set up for service at queues with higher inventory holding costs.

Buyukkoc, Varaiya and Walrand[1] show that if the server switches to other queues instantly without having any downtime ($D_{ij} = 0$, for each i and j, $i \neq j$), the optimal policy is to produce items with the shortest weighted mean processing time. The weighted mean processing times for jobs of type i is defined as $\frac{\mathbb{E}S_i}{c_i}$.
When the service times are exponentially distributed, this well celebrated policy is called the $c\mu$-rule, because the server is said to earn rewards at a rate of $c_i\mu_i$, when it serves a queue with weighted mean processing time $(c_i\mu_i)^{-1}$.

In polling systems the set-up times are usually strictly positive, in which case the $c\mu$-rule is not necessarily optimal. The complete characterization of an optimal policy has not been found so far, however heuristics for this problem have been developed. Van Oyen and Duenyas[8] considered this very problem, found some characteristics of the optimal policy and provided a simple heuristic policy, which regulates all the service, idling and sequencing decisions. They renumber the queues such that $\frac{c_i}{\mathbb{E}S_i} \geq \frac{c_j}{\mathbb{E}S_j}$ if $i < j$. Now, the queues are sorted in ascending order of weighted mean processing time. When the server is processing jobs at queue 1, the server will serve this queue in an exhaustive manner till the queue is completely empty, since this queue has the shortest weighted mean processing time and thus yields the most urgency. Whenever the queue is empty, the server will consider to switch to lesser urgent queues. If

the server is currently set up for service at queue 2, it will process jobs at queue 2 and after each job evaluate whether a set-up to queue 1 is needed. When queue 2 becomes empty, the server will consider to switch to one of the higher numbered queues, if queue 1 is empty. In general, if the queue is processing jobs at queue j, then every time after finishing a job, it considers switching to a queue i if $1 \leq i < j$. If queue j becomes empty, it also considers switching to queue k if $j \leq k < n$.

The exact rules of the heuristic are as follows. Assume that the server is set up to serve queue i and queue i contains x_i jobs. Furthermore, renumber the queues such that $\frac{c_1}{\mathbb{E}S_1} \geq \frac{c_2}{\mathbb{E}S_2} \geq ... \geq \frac{c_n}{\mathbb{E}S_n}$. We define

$$\phi_j(x_j) = \frac{c_j(x_j + \frac{\mathbb{E}D_j}{\mathbb{E}A_j})}{\mathbb{E}S_j(x_j + \frac{\mathbb{E}D_j}{\mathbb{E}S_j})}$$

and

$$\varphi_j(x_j) = \frac{c_j(x_j + \frac{\mathbb{E}D_j}{\mathbb{E}A_j})}{\mathbb{E}S_j(x_j + \frac{\mathbb{E}D_j}{\mathbb{E}S_j} + (\frac{1}{\mathbb{E}S_j} - \frac{1}{\mathbb{E}A_j})\mathbb{E}D_i)}$$

- If $x_i = 0$, then
  1. Let $\sigma = \emptyset$.
  2. For all $j \neq i$, if $\phi(x_j) \geq \frac{c_j\rho_j}{\mathbb{E}S_j}$, then let $\sigma = \sigma \cup j$.
  3. If $\sigma \neq \emptyset$, then among all $j \in \sigma$, let k denote the queue such that $k = \text{argmax}_{j\in\sigma}\, \phi_j(x_j)$. If $x_k \geq \frac{\mathbb{E}D_i}{\mathbb{E}A_k}$, then switch to queue k, else idle until the next arrival to the system.
  4. If $\sigma = \emptyset$, then let k denote the queue such that $k = \text{argmax}_{j\neq i}\, \phi(x_j)$. If $x_k > \frac{\mathbb{E}D_i}{\mathbb{E}A_k}$, then switch to queue k else idle until the next arrival to the system.
- If $x_i > 0$, then
  1. Let $\sigma = \emptyset$
  2. For all $j < i$, compute $\varphi_j(x_j)$. For all $j < i$, if for queue j $\varphi_j(x_j) \geq \frac{c_j\rho}{\mathbb{E}S_j} + \frac{c_i(1-\rho)}{\mathbb{E}S_i}$, then $\sigma = \sigma \cup j$.
  3. Among all $j \in \sigma$, switch to the queue that has the highest index $\varphi_j(x_j)$. If σ is empty ,then serve one more job of type i.

For the general idea and intuition behind these rules, see [8]. Van Oyen and Duenyas[8] show by means a simulation study that this heuristic outperforms several other policies in terms of total holding costs in situations with symmetric and asymmetric queues, high and moderate utilization, and both equal and different holding costs for different job classes. Moreover, this heuristic is easy to implement in systems, which makes it a very interesting heuristic in polling models where holding cost rates are not equal for each queue.
Duenyas[2] states that this heuristic is not subject to the variance effects described in section 2.2 and provides a simulation example with two queues which supports his statement. In the next chapter, we will also expose this heuristic to a simulation study using a scenario with more than two queues.

Chapter 3

Simulation

The majority of the in chapter 2 described policies defy an exact analysis, which means there are no closed-form expressions available to compute the overall and marginal queue length distributions. One has to resort to discrete time event simulation to shed some light on the behaviour of the queue lengths in these cases. In discrete time event simulation the operation of a system is represented as a chronical sequence of events. In the case of polling systems such events consist of things like job arrivals, the start of job processings and the start of set-up periods. Of each event a system time is recorded.

The simulation is initiated by creating the event for the first job arrival at each queue. The system time for all of these events (in the case of job arrivals, the time of the arrival) are determined by drawing numbers out of the interarrival time distribution. After that, all events will be 'handled' in a chronological order: each event will result in new events to be added to the chronical sequence[1], all in accordance to the specified service-, sequencing- and idling policies. Of course, these resulting events will be 'handled' as well at a later point in time, which will result in even more events to be added. During this process of handling events and creating new events a log file can be kept of the history of the systems behaviour, from which certain system characteristics like the average lengths of each separate queue can be deduced, simply by looking at the past events. Another option is to get this information 'on-the-fly', which means that while handling the events information about the average queue lengths is updated continuously.
After the system time of the events has grown long enough, the process of handling events can be stopped and a reliable time weighted average for certain specifics of the system can be calculated. Now, estimations of the performance measures are derived without the help of complicated formulas or expressions.
Often the first part of the simulation period is regarded as a 'warm-up'-time, in which the system is not in a steady-state situation yet. In that case, the time weighted average does not consider this period of time.

Of course, there are several drawbacks to this method. It cannot be done by hand, because it is simply not doable for a human being to perform a vast amount of such computations consecutively. The computation power of a computer is needed and even then the simula-

[1]For example, if a job arrives at a certain queue, an event for the next job arrival at that queue is scheduled and added to the sequence at the right place.

tions can take up quite a bit of time. Also, simulation does not always guarantee a reliable outcome. When numbers are drawn from distribution of which moments are not necessarily finite, the outcomes can be horribly unreliable.
On the other hand, it sometimes brings the only possibility of evaluation of systems which would otherwise be impossible. This is the case in our situation.

Solely for the purpose of studying the variance effects of the set-up times when adhering to several different policies, a discrete time simulation tool was developed. Afterwards, several scenarios were evaluated and its outcomes are presented in this chapter. Each numerical result found in this chapter was a result obtained from a simulation run of 2,000,000 units of time, of which the first 500,000 were regarded as warm-up time. Of each result only two positions after the decimal point is given.
In the different scenarios the same distribution was used for the duration of any set-up period, and is described in appendix A. This distribution has the property that by raising its parameter u, the expectation will fall considerably, while the ratio of the variance versus the expectation tips over in favour of the variance more and more. These characteristics are exactly the ones that should provoke the variance effect described in section 2.2.
Moreover, when comparing the different policies, we used one different policy to those used in the original scenario (see section 2.2) at the time. All the other policies and possibly relevant variables we kept constant as much as possible, as to rule out possibilities of mutually deviating results being caused by anything else than the difference in the one policy.
Recall that the observation of increasing (decreasing) waiting times in a certain queue is equivalent to the observation of an increasing (decreasing) mean length of that queue, through Little's law.

3.1 Simulation study of service policies

To see whether the set-up variance effect occurs with other service policies than the exhaustive service policy, a polling system with two queues is considered. We assume that the arrival processes are Poisson with rates $\lambda_1 = 2$ and $\lambda_2 = 3$. The service times are exponentially distributed with rates $\mu_1 = 10$ and $\mu_2 = 12$ respectively.
The queues have an infinite buffer capacity, which means that there is always space for a job to wait at its queue. For all set-up distributions D_{ij} the distribution described in appendix A is used. A no-idling policy and a cyclic sequencing policy is assumed, as is the case in Zangwill's example (see section 2.2).

We observe the behaviour of the system under the exhaustive service policy, the quantity-limited service policy and the time-limited service policy. We know already that the exhaustive service policy should exhibit the variance effect. For observing the behaviour of the system under the quantity-limited policy, the limits $k_1 = 7$ and $k_2 = 11$ were chosen. In case $u = 2$, $ED = \frac{30}{16}$ and the system is only just stable for these limits. Indeed, when we check condition 2.2, the inequality holds by a very small margin:

$$\rho + \mathbb{E}D \max_{i=1,\ldots,n} \frac{\lambda_i}{k_i} = (\frac{2}{10} + \frac{3}{12}) + \frac{30}{16} \max\{\frac{2}{7}, \frac{3}{11}\} = \frac{138}{140} < 1$$

For the time-limited policy, the limits $t_1 = \frac{7}{10}$ and $t_2 = \frac{11}{12}$ are used. When $u = 2$ the system is stable, but again near the edge of unstability.

	Exhaustive			Quantity-Limited			Time-Limited		
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W$	$\mathbb{E}W_1$	EW_2	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W$
2	2.17	2.05	2.10	47.41	12.38	26.40	9.23	7.90	8.43
3	2.10	2.01	2.05	3.22	2.88	3.02	3.01	2.84	2.91
4	2.40	2.31	2.34	3.01	2.81	2.89	2.89	2.81	2.84
5	2.80	2.69	2.74	3.24	3.07	3.14	3.15	3.08	3.11
6	3.26	3.16	3.20	3.62	3.45	3.52	3.53	3.46	3.49
7	3.73	3.59	3.64	4.04	3.87	3.94	3.92	3.88	3.90
8	4.24	4.08	4.15	4.50	4.32	4.39	4.41	4.35	4.37
9	4.70	4.53	4.60	4.95	4.76	4.84	4.85	4.81	4.82
10	5.21	5.02	5.10	5.43	5.22	5.31	5.31	5.27	5.29
15	7.75	7.48	7.59	7.88	7.62	7.72	7.73	7.70	7.71
25	12.89	12.44	12.62	12.90	12.49	12.65	12.66	12.68	12.68
50	25.32	24.59	24.89	25.49	24.66	24.99	25.08	25.18	25.14

Table 3.1: *Simulated mean waiting times using a no-idling policy, a cyclical sequencing policy and several service policies.*

Using different u-values, the simulation tool came up with the results shown in Table 3.1. A graphical representation of the behaviour of the average mean waiting times under the service policies can be found in Figure 3.1.

We see that in the case of $u = 2$, the waiting times under the quantity-limited policy and the time-limited policy are very high compared to the exhaustive policy. This is because of the near unstability of the system. When the expectation of the switch-over time reduces, the waiting times reduce significantly at first as well. While under the exhaustive policy the waiting times become shorter just a tiny bit when we change from $u = 2$ to $u = 3$, the variance effect becomes clear afterwards and the higher the u, the longer the waiting times become and thus the larger the queue lengths grow. The point of turnover in case of the quantity-limited and the time-limited policy is a bit later at $u = 4$. These policies contain a processing limit, and because of lowering the set-up times the system will move away from being near unstable[2] between $u = 2$ and $u = 4$. This effect makes the queue lengths go down for higher u. However, as u keeps increasing after $u = 4$, this stability effect vanishes and the limits of these policies do incur less additional switch-over time over the exhaustive policy, since the expectation of the duration of the set-up periods decrease. As a consequence the queue lengths become less responsive to the service policy, because the distinction between the service policies decreases. As such the variance effect also shows itself using the quantity-limited and time-limited policy.

There is a special case of a polling system using the quantity-limited policy known for which analysis is tractable. The above observed queue length behaviour can also be shown analytically in this case, as we will see in the next example.

Example 3.1 *We consider a polling system, in which the server adheres to a 1-limited service policy, never idles and switches through the queues in a cyclical manner. The polling*

[2] $\mathbb{E}D$ reduces, so in case of the quantity-limited policy inequality 2.2 is more likely to hold. Something similar goes for the time-limited policy.

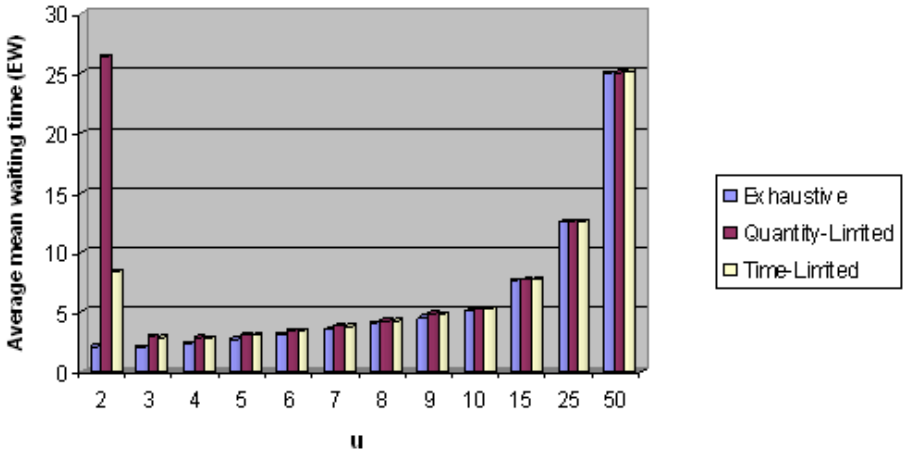


Figure 3.1: *The behaviour of the average mean waiting time under a cyclical sequencing policy, a no-idling policy and several service policies.*

system is fully symmetric, which means that all queues are equal in terms of arrival processes and service time distributions and that the duration distributions of all possible set-up periods are identical to each other.

Winands[12] shows that the following equation holds for this particular system:

$$\sum_{i=1}^{n} \rho_i \mathbb{E}W_i = \rho \sum_{i=1}^{n} \frac{\lambda_i \mathbb{E}(S_i^2)}{2(1-\rho)} + \rho \frac{Var(D)}{2\mathbb{E}D} + \frac{\mathbb{E}D}{2(1-\rho)} \sum_{i=1}^{n} \rho_i(1-\rho_i) + \sum_{i=1}^{n} \rho_i \lambda_i \frac{\mathbb{E}D}{1-\rho}(\mathbb{E}W_i + ES_i)$$

where D is again a random variable denoting the total duration of the set-up periods in one cycle.
When we use the fact that $\rho = \sum_{i=1}^{n} \rho_i$, we can reduce this equation to an expression for the expected waiting times $\mathbb{E}W_i$ (i=1, ..., n):

$$\mathbb{E}W_i = \frac{\rho \sum_{i=1}^{n} \frac{\lambda_i \mathbb{E}(S_i^2)}{2(1-\rho)} + \rho \frac{Var(D)}{2\mathbb{E}D} + \frac{\mathbb{E}D}{2(1-\rho)} \sum_{i=1}^{n} \rho_i(1-\rho_i) + \rho \lambda_i \frac{\mathbb{E}D}{1-\rho} \mathbb{E}S_i}{\rho(1 - \lambda_i \frac{\mathbb{E}D}{1-\rho})} \tag{3.1}$$

Of course, since the polling system is symmetric, the expectations of the waiting times in the different queues are the same. We see that equation 3.1 has potential for the variance effect to occur because of the term $\rho \frac{Var(D)}{2\mathbb{E}D}$ in the nominator.

Let us consider this system with two queues, each with a Poisson arrival process, each having a rate of $\lambda_i = 0.3$, and exponentially distributed service times with expectation $\mathbb{E}S_i = 0.725$. The duration of the set-up periods are all distributed according to the random variable described in appendix A with parameter u. In case $u = 2$, the system is only just stable. When checking condition 2.2, we find

$$\rho + \mathbb{E}D \max_{i=1,\ldots,n} \frac{\lambda_i}{k_i} = \frac{174}{400} + \frac{30}{16} * \frac{3}{10} = \frac{399}{400} < 1$$

and see that it indeed only just holds. Now, when we compute $\mathbb{E}W_i$ with the available information and equation 3.1, we see that in case of $u = 2$ the waiting times are huge with an expected duration of 744.21. This high number is a result of the denominator $\rho(1 - \lambda_i \frac{\mathbb{E}D}{1-\rho})$, which in this case amounts to a number of 0.001925.
When we increase u to 3, the denominator rises to 0.2334932 and consequently the expected waiting time falls immensely to 5.33. This is the very same effect as the effect we have called the stability effect earlier.
When we increase u further, the denominator will keep increasing, however it cannot grow beyond $\rho = 0.6$. Instead, it will converge to this value. Hence the denominator will be increasing at a decreasing rate, which means that the stability effect will grow feebler and feebler. When $u = 5$, the expected waiting time will become 4.03. However, when u increases any further, the relative increments of the nominator through the term $\rho \frac{Var(D)}{2\mathbb{E}D}$ will be higher than those of the denominator. When $u = 6$, the average queue length has risen to 4.26 and will continue to rise as u keeps increasing. The variance effect has grown stronger than the stability effect.

3.2 Simulation study of sequencing policies

After studying the service policies, we wish to examine the behaviour of the in subsection 2.4 described sequencing policies. To do this, we consider a polling system with three queues. The consideration of a system with more than two queues is needed, since otherwise all sequencing policies would be the same. In case of two queues, the only option when the server is at queue 1 would be to set up to queue 2 and vice versa.

All queues have a Poisson arrival process, with rates $\lambda_1 = 2$, $\lambda_2 = 1$ and $\lambda_3 = 3$. The service times are deterministic with $\mathbb{E}S_1 = 0.2$, $\mathbb{E}S_2 = 0.25$ and $\mathbb{E}S_3 = 0.05$ respectively. All the queues have infinite capacity, there is always enough space in the queue for an arriving job. For every possible set-up the distribution described in appendix A is assumed, all of them with the same parameter u. Furthermore, an exhaustive service policy and a no-idling policy is used. In the case of a cyclical sequencing policy, this scenario reduces to the scenario considered by Zangwill(see section 2.2) and hence the queue lengths will grow without bounds at least in that case.

We simulated the above described scenario combined with the following sequencing policies:

- Two pre-set sequencing policies:
  - (**Cyclical**) The cyclical sequencing policy (Zangwill's scenario)
  - (**Pre-set**) A pre-set sequencing policy with service order 1-2-1-3-2-3
- (**LQF**) The Longest-Queue-First policy
- Several policies prioritizing by characteristics:
  - (**PHVL**) The PHVL sequencing policy
  - (**PLEA**) The PLEA sequencing policy
  - (**PHES**) The PHES sequencing policy

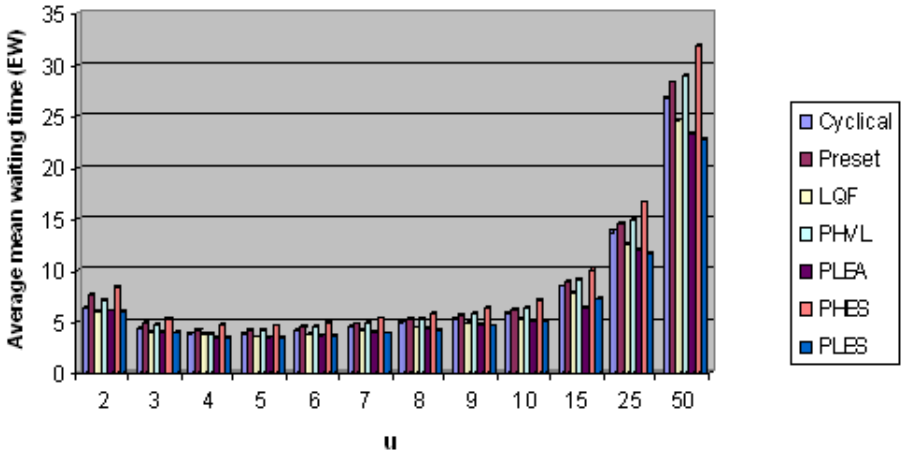


Figure 3.2: *The behaviour of the average mean waiting time under the exhaustive service policy, a no-idling policy and several sequencing policies.*

– (**PLES**) The PLES sequencing policy

The results of these simulations can be found in Table 3.2. A graph of the behaviour of the average mean waiting time $\mathbb{E}W$ is depicted in Figure 3.2.
We see that in case of the cyclical sequencing policy, the variance effect shows itself as expected. Initially, when increasing u to 4, the queue lengths go down, but afterwards the queue lengths seem to grow without bound. We also see that the variance effect also shows itself with every other considered sequencing policy; after the point of turnover of $u = 4$, the queue lengths seem to explode for every sequencing policy. Moreover, the queue lengths do not mutually deviate very much when using different sequencing policies.

Hence, it seems that the sequencing policy plays no role in the degree of existence of the variance effect whatsoever. This is not hard to believe, since whichever sequencing decision is taken, the ratio of the variance over the expectation of the distribution underlying the duration of the set-up period incurred grows without bound as u increases.

3.3 Simulation study of idling policies

For the study of the idling policies, the same polling model as in section 3.2 is considered. Again, we have three queues with Poisson arrival processes having rates $\lambda_1 = 2$, $\lambda_2 = 1$ and $\lambda_3 = 3$ respectively. The service times are deterministic as before with $\mathbb{E}S_1 = 0.2$, $\mathbb{E}S_2 = 0.25$ and $\mathbb{E}S_3 = 0.05$. The queues still have an infinite capacity buffer, so there is always enough space for an arriving job to wait at the queue. The set-up times are distributed according to the distribution described in appendix A with parameter u.

To study the existence of the variance effect with several idling policies, an exhaustive service policy is used. For the sequencing policy both cyclical and 'Longest-Queue-First' sequencing policies are used. We use the cyclical sequencing policy because of the fact this policy was used in Zangwill's example (section 2.2) as well, so any conclusions based on this study will

	Cyclical				Pre-set				(LQF)			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	5.20	6.48	7.35	6.49	5.60	6.39	9.37	7.62	5.65	7.81	5.98	6.17
3	3.50	4.36	4.89	4.34	3.69	4.26	6.04	4.96	3.61	5.19	3.85	3.99
4	3.14	3.92	4.36	3.88	3.20	3.73	5.13	4.25	3.14	3.92	4.36	3.88
5	3.17	3.96	4.35	3.89	3.24	3.80	5.11	4.27	3.18	4.52	3.41	3.52
6	3.42	4.29	4.69	4.20	3.44	4.07	5.39	4.52	3.42	4.82	3.67	3.78
7	3.70	4.66	5.07	4.54	3.72	4.38	5.78	4.86	3.77	5.19	4.01	4.13
8	4.08	5.13	5.56	4.99	4.06	4.83	6.27	5.29	4.12	5.60	4.36	4.49
9	4.44	5.59	6.03	5.43	4.45	5.28	6.68	5.79	4.56	6.15	4.83	4.96
10	4.86	6.13	6.63	5.96	4.83	5.70	7.41	6.26	4.99	6.63	5.27	5.40
15	6.97	8.76	9.43	8.50	6.91	8.18	10.51	8.93	7.73	9.43	7.63	7.80
25	11.38	14.26	15.32	13.83	11.17	13.27	17.10	14.49	11.93	15.01	12.31	12.64
50	22.13	27.75	29.69	26.85	21.76	26.09	33.26	28.23	23.26	29.08	23.81	24.51

	PHVL				PLEA				PHES			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	4.77	6.09	9.01	7.11	5.93	7.75	5.89	6.21	4.54	5.67	11.96	8.44
3	3.14	4.17	5.97	4.73	3.86	5.31	3.64	3.99	3.12	3.65	7.52	5.41
4	2.61	3.55	4.76	3.84	3.35	4.67	3.11	3.45	2.84	3.24	6.48	4.73
5	2.87	3.86	5.30	4.25	3.36	4.74	3.12	3.47	2.90	3.26	6.37	4.69
6	3.10	4.16	5.69	4.57	3.57	5.06	3.31	3.69	3.15	3.51	6.78	5.03
7	3.37	4.54	6.15	4.96	3.81	5.42	3.55	3.95	3.43	3.83	7.33	5.45
8	3.69	4.99	6.71	5.41	4.23	6.02	3.94	4.38	3.75	4.17	7.95	5.92
9	4.03	5.44	7.28	5.89	4.58	6.53	4.27	4.75	4.09	4.57	8.67	6.46
10	4.41	5.97	7.96	6.44	4.97	7.10	4.65	5.16	4.49	5.01	9.49	7.08
15	6.36	8.68	11.33	9.24	5.20	6.48	7.35	6.49	6.47	7.19	13.49	10.10
25	10.29	13.95	18.26	14.89	11.5	16.68	10.69	11.99	10.48	11.75	21.83	16.73
50	20.11	27.48	35.21	28.88	22.58	32.81	20.72	23.35	20.29	23.06	42.26	31.74

	PLES			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	6.67	8.33	5.00	6.12
3	4.26	5.41	3.30	3.97
4	3.56	4.79	2.89	3.43
5	3.49	4.82	2.89	3.42
6	3.67	5.13	3.08	3.62
7	3.96	5.57	3.34	3.92
8	4.28	6.04	3.64	4.25
9	4.70	6.63	3.99	4.67
10	5.10	7.23	4.35	5.08
15	7.22	10.34	6.17	7.21
25	11.72	16.80	9.99	11.70
50	22.70	33.06	19.25	22.70

Table 3.2: *Simulated mean waiting times using an exhaustive service policy, a no-idling policy and several sequencing policies.*

not be caused by a different sequencing policy. The LQF sequencing policy has to be used, because when evaluating the behaviour of the 'Switch to important queue first'-policy, one needs to assume complete freedom of set-up for the server from any queue i to any queue j. This assumption is violated by the cyclical sequencing policy. In section 3.2 we saw that in this polling system the sequencing policy does not impact the existence of the variance effect in any way, which justifies the use of an LQF-sequencing policy.

First we study the situation in which the cyclical sequencing policy is used. We study the behaviour of the no-idling policy and the empty-system-idling policy. As done before, we examine the queue lengths of the system with different values of u. The higher u, the lower the expectation of the duration of the set up times, but the higher the variance becomes relative to the expectation.
Afterwards we will study the behaviour of all the in section 2.5 described idling policies under the LQF sequencing policy in very much the same way.

3.3.1 Idling policies under the cyclical sequencing policy

When under the exhaustive service policy and cyclical sequencing policy simulations are run, the results are very interesting. These results can be found in Table 3.3. In Figure 3.3, the behaviour of the average mean waiting time is shown, based upon the results of the table. In case of the no-idling policy, our situation reduces to the circumstances in which Zangwill's analysis holds and indeed the variance effect of the set-up time is clearly noticeable. The overall queue length and the average queue lengths go down initially when u is raised from 2 to 4, but afterwards the variance effect gets the leading hand and the queue lengths grow without bound. When the empty-system-idling policy is used however, *there is no noticeable variance effect*! When we keep increasing u, both the marginal and overall queue lengths keep decreasing. Even in case of $u = 50$ these queue lengths seem to have become smaller. In the transition from $u = 2$ to $u = 50$, the expected duration of a set-up period has gone down from $\frac{15}{16}$ to approximately $\frac{1}{625}$, while the ratio of $\frac{\mathrm{Var}(D)}{\mathbb{E}D}$ has gone up from $\frac{343}{240}$ to approximately 50. Of course, one could be in denial of the non-existence of the effect and state that it still might occur with a much higher u. However, we see here that the transition considered is a huge one. Moreover, in practice set-up time cuts like these or even greater ones will not be achieved, so we think it is safe to postulate that the variance effect does not show itself in this polling system.
Srinivasan and Gupta[11] show that in general the reduction of set-up time actually can increase queue lengths in certain situations using empty-system-idling. In other words, it is not always the case that the queue lengths will decrease monotonously while set-up times are cut. However, based on the simulation results, it seems very unlikely that the queues can grow without bound when an empty-system-idling policy is used.

3.3.2 Idling policies under the LQF sequencing policy

Under the LQF sequencing policy, we consider the no-idling policy, empty-system-idling policy and several switch-to-important-queue-first policies. These include the following:

	No-idling				Empty-system-idling			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	5.20	6.48	7.35	6.49	5.14	6.43	7.28	6.43
3	3.50	4.36	4.89	4.34	3.48	4.33	4.86	4.31
4	3.14	3.92	4.36	3.88	3.03	3.79	4.21	3.74
5	3.17	3.96	4.35	3.89	2.85	3.56	3.93	3.51
6	3.42	4.29	4.69	4.20	2.76	3.44	3.78	3.38
7	3.70	4.66	5.07	4.54	2.69	3.37	3.69	3.30
8	4.08	5.13	5.56	4.99	2.56	3.21	3.51	3.14
9	4.44	5.59	6.03	5.43	2.44	3.06	3.37	3.01
10	4.86	6.13	6.63	5.96	2.36	2.95	3.25	2.90
15	6.97	8.76	9.43	8.50	1.87	2.33	2.56	2.29
25	11.38	14.26	15.32	13.83	1.32	1.64	1.84	1.63
50	22.13	27.75	29.69	26.85	0.74	0.85	1.03	0.90

Table 3.3: *Simulated mean waiting times using an exhaustive service policy, a cyclical sequencing policy and several idling policies.*

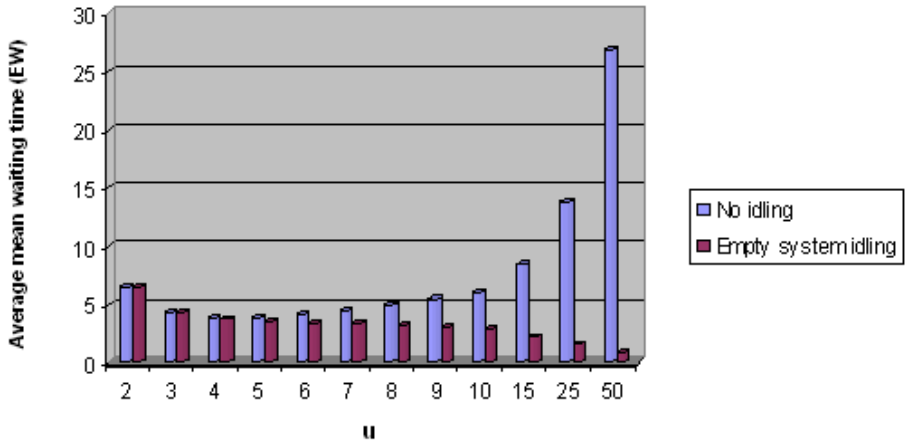


Figure 3.3: *The behaviour of the average mean waiting time under the cyclical sequencing policy and several idling policies.*

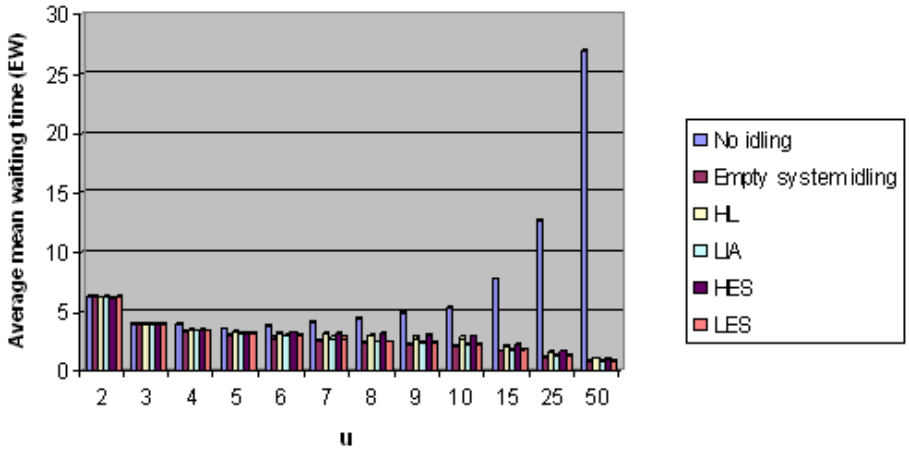


Figure 3.4: *The behaviour of the average mean waiting time under the exhaustive service policy, LQF sequencing policy and several idling policies.*

- Highest Load (HL): Whenever the system becomes empty, the server will switch to the queue with the highest load, before it starts an idling period. In this case, the queue with the highest load is queue 1.

- Lowest expected interarrival time (LIA): Whenever the system becomes empty, the server will switch to the queue having the lowest expected interarrival time first, before it starts an idling period. In this case, the queue with the lowest expected interarrival time is queue 3.

- Highest expected service time (HES): Whenever the system becomes empty, the server will switch to the queue having the highest expected service time first, before it starts an idling period. In this case, the queue with the highest expected service time is queue 2.

- Lowest expected service time (LES): Whenever the system becomes empty, the server will switch to the queue having the lowest expected service time first, before it starts an idling period. In this case, the queue with the lowest expected service time is queue 3.

In this situation, the LIA- and LES-policies are completely identical.

For each of the examined idling policies, simulations have been run and the results of these can be found in Table 3.4. A graph depicting the behaviour of the average mean waiting time under these policies can be found in Figure 3.4. As we see, the variance effect shows itself in the case of the no-idling policy as already concluded in section 3.2, but it does not while using any other of the idling policies in this particular polling system! On a side note, we see that the switch-to-important-queue-first idling policies do not perform as well as the empty-system-idling policy, in both marginal queue lengths and overall queue length, also in case of high u's.

	No-idling				Empty-system-idling				HL			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	5.65	7.81	5.98	6.17	5.62	7.80	5.96	6.26	5.63	7.80	5.95	6.15
3	3.61	5.19	3.85	3.99	3.58	5.16	3.81	3.96	3.60	5.18	3.83	3.98
4	3.14	3.92	4.36	3.88	2.97	4.31	3.18	3.30	3.09	4.50	3.31	3.44
5	3.18	4.52	3.41	3.52	2.71	3.94	2.90	3.01	2.88	4.26	3.10	3.22
6	3.41	4.82	3.67	3.78	2.54	3.70	2.71	2.82	2.81	4.18	3.03	3.15
7	3.77	5.19	4.01	4.13	2.38	3.49	2.55	2.65	2.75	4.12	2.96	3.08
8	4.12	5.60	4.36	4.49	2.24	3.28	2.39	2.49	2.61	3.92	2.84	2.94
9	4.56	6.15	4.83	4.96	2.08	3.05	2.21	2.31	2.54	3.80	2.74	2.85
10	4.99	6.63	5.27	5.40	1.95	2.85	2.10	2.17	2.46	3.71	2.68	2.78
15	7.25	9.43	7.63	7.80	1.54	2.23	1.65	1.71	1.95	2.95	2.11	2.19
25	11.93	15.01	12.31	12.64	1.08	1.57	1.16	1.20	1.42	2.11	1.53	1.59
50	22.13	27.75	29.69	26.85	0.67	0.97	0.77	0.77	0.94	1.39	1.08	1.08
	LIA				HES				LES			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	5.65	7.80	5.97	6.17	6.66	8.33	4.99	6.10	5.65	7.80	5.97	6.17
3	3.58	5.14	3.82	3.96	3.59	5.15	3.82	3.96	3.58	5.14	3.82	3.96
4	3.03	4.41	3.24	3.37	3.09	4.46	3.29	3.42	3.03	4.41	3.24	3.37
5	2.77	4.04	2.95	3.07	3.17	4.36	2.63	3.10	2.77	4.04	2.95	3.07
6	2.62	3.83	2.79	2.90	2.97	4.13	3.06	3.21	2.62	3.83	2.79	2.90
7	2.47	3.64	2.63	2.75	2.90	4.02	2.96	3.11	2.47	3.64	2.63	2.75
8	2.30	3.37	2.45	2.55	2.86	3.96	2.91	3.07	2.30	3.37	2.45	2.55
9	2.23	3.26	2.37	2.47	2.79	3.82	2.81	2.97	2.23	3.26	2.37	2.47
10	2.07	3.04	2.21	2.30	2.67	3.68	2.71	2.86	2.07	3.04	2.21	2.30
15	1.67	2.43	1.76	1.84	2.14	2.91	2.14	2.27	1.67	2.43	1.76	1.84
25	1.15	1.68	1.25	1.29	1.63	2.19	1.63	1.72	1.15	1.68	1.25	1.29
50	0.67	1.01	0.78	0.78	0.95	1.30	1.01	1.04	0.67	1.01	0.78	0.78

Table 3.4: *Simulated mean waiting times using an exhaustive service policy, an LQF sequencing policy and several idling policies.*

3.3.3 Possible explanations

We have seen that both under the cyclical sequencing policy and the LQF sequencing policy, any idling policy apart from the no-idling policy do not allow the queues to grow without bound. An explanation for these observation might be that the server will be idling more when set-up times are cut in case the empty-system-idling policy is used, whereas in case of the no-idling policy the server keeps switching and incurring set-up times. Because a lot less set-up time is used overall, the variance effect might be too feeble to persist or even show itself. Then again, this is pure speculation as analysis of systems using empty-system-idling policies is hardly tractable.

In literature it is shown that idling can be employed to counter the variance effect. Van Oyen[7] says that when using idling, the systems performance can at least not degrade if set-up times are cut. When set-up times are cut, one can always decide to lengthen the set-up periods with an idling period such that the sum of the duration of these periods equals the duration of an original set-up period. Then, the queue lengths will not have changed. Van Oyen then says that this particular idling policy may be far from optimal and that as such the optimal policy using a similar form of idling will at least perform as well and maybe even better. While the form of idling we have studied, such as empty-system-idling, does not exactly work in the same way (it is not used to artificially increase the duration of set-up periods), it gives another insight in how idling can be seen as a means to prevent the variance effect to occur.

3.4 Simulation study of the heuristic by Van Oyen and Duenyas

The policy we study in this section is the heuristic by Van Oyen and Duenyas as explained in section 2.6. As we already know, this heuristic constitutes a service policy, a sequencing policy and an idling policy. We consider the same polling system as in the previous two sections with three queues, since a sequencing decision is involved. Once again, the queues of this system has Poisson arrival processes with rates $\lambda_1 = 2$, $\lambda_2 = 1$ and $\lambda_3 = 3$. The service times are deterministic and have durations $\mathbb{E}S_1 = 0.2$, $\mathbb{E}S_2 = 0.25$ and $\mathbb{E}S_3 = 0.05$ respectively. The distribution of the durations of all possible set-up periods is the one described in appendix A with parameter u.
Within these specifications, we consider the system with three different holding cost settings:

- $c_1 = 0.2$, $c_2 = 0.25$ and $c_3 = 0.05$. We choose these holding cost rates, because then the weighted mean processing times are all equal: $\frac{\mathbb{E}S_1}{c_1} = \frac{\mathbb{E}S_2}{c_2} = \frac{\mathbb{E}S_3}{c_3} = 1$.
- $c_1 = c_2 = c_3 = 1$. These settings treat inventory in every queue as equally undesirable. The weighted mean processing times are $\frac{\mathbb{E}S_1}{c_1} = 0.2$, $\frac{\mathbb{E}S_2}{c_2} = 0.25$ and $\frac{\mathbb{E}S_3}{c_3} = 0.05$.
- $c_1 = 3$, $c_2 = 5$ and $c_3 = 7$. These settings treat inventory in queue 3 as more undesirable than inventory in queue 2, and inventory in queue 2 on its turn more undesirable than inventory in queue 1. The weighted mean processing times in this case are $\frac{\mathbb{E}S_1}{c_1} = \frac{1}{15}$, $\frac{\mathbb{E}S_2}{c_2} = \frac{1}{20}$ and $\frac{\mathbb{E}S_3}{c_3} = \frac{1}{140}$.

The results of the simulations performed can be found in Table 3.5. The behaviour of the average mean waiting time is depicted in Figure 3.5. We see that the average queue length

	$c_1 = 0.2, c_2 = 0.25, c_3 = 0.05$				$c_1 = 1, c_2 = 1, c_3 = 1$				$c_1 = 3, c_2 = 5, c_3 = 7$			
u	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	EW_2	$\mathbb{E}W_3$	$\mathbb{E}W$	$\mathbb{E}W_1$	$\mathbb{E}W_2$	$\mathbb{E}W_3$	$\mathbb{E}W$
2	4.14	6.39	10.20	7.54	4.43	8.00	8.41	7.02	5.10	6.51	8.00	6.78
3	2.55	5.32	5.71	4.59	3.12	5.50	4.60	4.26	3.74	4.25	4.63	4.27
4	2.39	3.58	5.48	4.13	2.73	5.11	3.27	3.40	3.83	3.37	3.29	3.48
5	2.22	3.28	4.91	3.74	2.37	5.51	2.40	2.91	4.32	2.60	2.32	3.04
6	2.11	3.12	4.55	3.50	2.09	5.65	1.85	2.56	4.46	2.15	1.78	2.73
7	1.98	2.94	4.25	3.27	1.82	5.77	1.53	2.33	4.56	1.74	1.49	2.55
8	1.88	2.78	3.98	3.08	1.66	5.92	1.29	2.18	4.58	1.55	1.26	2.41
9	1.75	2.58	3.70	2.86	1.53	5.56	1.10	1.99	4.40	1.40	1.08	2.24
10	1.66	2.43	3.47	2.69	1.46	5.66	0.98	1.92	4.09	1.27	0.94	2.04
15	1.31	1.93	2.77	2.14	1.12	4.74	0.65	1.49	3.73	0.96	0.61	1.71
25	0.98	1.44	2.08	1.60	0.83	3.82	0.43	1.13	3.59	0.75	0.46	1.55
50	0.65	0.92	1.42	1.08	0.63	2.72	0.27	0.80	1.81	0.42	0.25	0.80

Table 3.5: *Simulated mean waiting times using the heuristical policy by Van Oyen and Duenyas.*

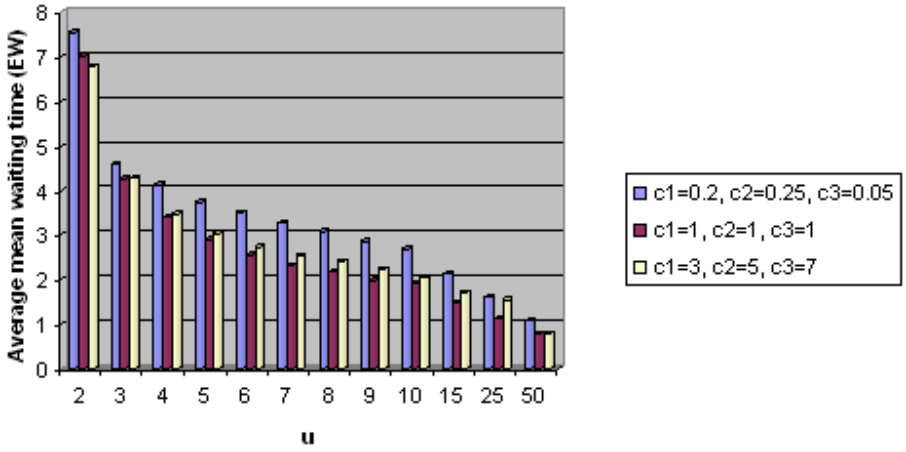


Figure 3.5: *The behaviour of the average mean waiting time under the heuristical policy by Van Oyen and Duenyas.*

decreases as u increases. It is hard to pinpoint what variable exactly is responsible for this, since this heuristical policy constitutes a service policy, an idling policy and a sequencing policy. One could speculate that the heuristical policy includes idling periods and, since we found before that idling policies other than the no-idling policy battle the variance effect, that this idling part makes it such that the queue lengths decrease.

Also, we see that most of the marginal queue lengths decrease as u increases. Moreover, when we look at the situations where the weighted mean processing times are not equal at every queue (the cases where $c_1 = c_2 = c_3 = 1$ and where $c_1 = 3, c_2 = 5, c_3 = 7$), we see that the lengths of the queues with the lowest weighted mean processing time can increase when u goes up! Indeed, when $c_1 = c_2 = c_3 = 1$ we see that the length of queue 2 decreases as u goes from 2 to 4, but then increases once again as u goes from 4 to 8. Afterwards, it decreases again. The same effect we see with the length of queue 1 when the holding cost rates $c_1 = 3$, $c_2 = 5$ and $c_3 = 7$ are used. As u goes from 2 to 3, the queue length goes down, however when u increases further to 8, the queue length increases! Afterwards, the queue length decreases again. One could speculate this observation to be a combination of several effects. We saw earlier in section 3.1 that a stability effect may account for the early reduction in queue length. An equivalent effect might be showing itself here. When $u = 2$, the length of queue 2 in case of $c_1 = c_2 = c_3 = 1$ and the length of queue 1 in case of $c_1 = 3, c_2 = 5, c_3 = 7$ are higher than what they would have been in case of $c_1 = 0.2, c_2 = 0.25, c_3 = 0.05$. These queues might be near instability: the server utilization is quite high (0.80) and the weird behaving queues themselves do not have any priority at all when it comes to being served. Furthermore, the server will be very reluctant to idle at these queues when they are not prioritized. Hence, because of the little idling, the variance effect might show itself for these queues and the marginal queue length goes up, when the stability effect has passed. If u then keeps increasing, the server will also consider idling at these queues, and the variance effect will be too feeble to let the queue length grow without bound.

Although this reasoning is pure speculation and the effects observed are not crystal clear to understand, we can conclude that using this policy, the variance effect does not present itself in such a way that the queue lengths grow without bound. Duenyas [2] already concluded this in case of a system with two queues.

3.5 The role of the set-up time distribution

Throughout the previous sections, we have used a very convenient distribution for the set-up times (see appendix A). With a change of one parameter, the variance effects were triggered if the policies used enabled it to occur. Not every distribution family is likely to exhibit the variance effect, consider example 3.2.

Example 3.2 *In case the duration of every set-up period can be modeled as an exponential distribution with a certain parameter λ, the variance effect is not likely to occur. Let X denote the duration of a set-up period. Then, $\mathbb{E}X = \lambda^{-1}$ and $Var(X) = \lambda^{-2}$. When set-up times are cut, $\mathbb{E}X$ is cut, and hence λ becomes bigger. However, $\frac{Var(X)}{\mathbb{E}X} = \frac{\lambda^{-2}}{\lambda^{-1}} = \lambda^{-1}$, which is also decreasing in λ! The variance decreases at a bigger rate than the expectation itself. Hence, Zangwill's observed variance effect will most likely not occur.*

a	$\mathbb{E}W_1$	EW_2	EW
0	0.16	0.15	0.15
0.1	0.30	0.28	0.29
0.2	0.46	0.42	0.43
0.3	0.63	0.57	0.59
0.4	0.82	0.73	0.76
0.5	1.07	0.91	0.98
0.6	1.42	1.15	1.26
0.7	2.04	1.51	1.73
0.8	3.68	2.24	2.82

Table 3.6: *The behaviour of the average mean waiting time under a quantity-limited sequencing policy and a no-idling policy, assuming uniformly distributed set-up durations.*

One could argue that the set-up time distributions considered both in the previous sections and this example are not realistic. The degree of variation of the distribution in Appendix A might be ridiculously high, while a memoryless distribution does not seem likely either.
We do not aim to research how set-up periods are generally distributed in practice, since that falls out of the scope of this paper. However, we do want to point out in this section that the set-up time distributions themselves must yield a high degree of variation relative to their expectation for the variance effects to occur. Otherwise, the increase of the term $\frac{\text{Var}(D_i)}{\mathbb{E}D_i}$ may be too feeble for every i to be able to exhibit the variance effects. While the distribution of Appendix A satisfies this condition amply, in practice the distribution might not have this property.

To illustrate this case further, let us consider a polling system with two queues. The arrival processes are Poisson with rates $\lambda_1 = 2$ and $\lambda_2 = 3$. The service times are also exponentially distributed with rates $\mu_1 = 10$ and $\mu_2 = 12$. This time, we do not consider the set-up distribution described in appendix A, the duration of a set-up period between any pair of queues is now uniformly distributed with parameters a and b, $b = a + 0.1$, $a > 0$. The variance of the duration of a set-up period is quite moderate in this case, it is equal to $\frac{0.1^2}{12} = \frac{1}{1200}$. This number is insensitive to the a-parameter, as this parameter only controls the location of the distribution, and hence only is positively correlated with the distributions expectation. So, when lowering a, the variance effect could occur theoretically, since the term $\frac{\text{Var}(D_i)}{\mathbb{E}D_i}$ rises for every i.
We assume a quantity-limited service policy with limits $k_1 = 7$ and $k_2 = 11$ and a no-idling policy. We simulated this scenario using several values of a. The results can be found in Table 3.6. A depiction of the behaviour of the average mean waiting time can be found in Figure 3.6. We see that as a decreases to 0 (it cannot go any lower), the queue lengths all monotonously decrease as well! We have shown however in section 3.1 that this particular combination of policies does allow variance effects to occur. For a variance effect to occur in this situation, one should increase the variance of the distribution itself, which is something that is absolutely not suggested by the Japanese production theory.

In case of a polling system with an exhaustive service policy, a cyclical sequencing policy

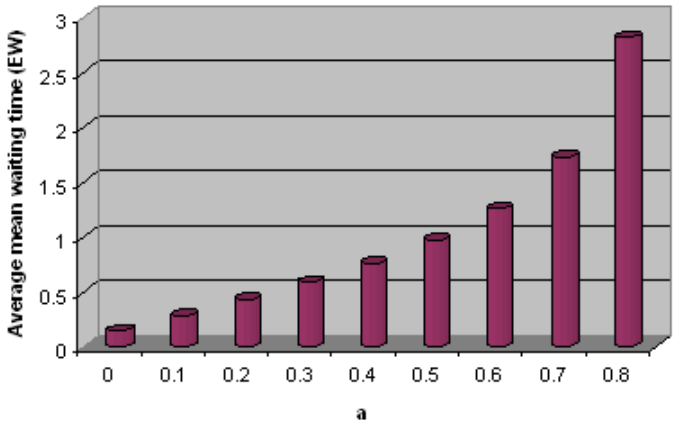


Figure 3.6: *The behaviour of the average mean waiting time under a quantity-limited sequencing policy and a no-idling policy, assuming uniformly distributed set-up durations.*

and a no-idling policy, we can also illustrate this matter analytically. In this case, equation 2.1 for the expected waiting time of queue i holds. We assume that $D_1 = D_2 = ... = D_n = D$. In other words, we assume that the duration of any set-up period is distributed according to a random variable D, regardless of the departure queue and destination queue. In that case, equation 2.1 can be simplified to

$$\begin{aligned} \mathbb{E}W_i &= \frac{1-\rho_i}{2}\frac{n\mathbb{E}D}{1-\rho} + \frac{1-\rho_i}{2}\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)\{\lambda_j\mathbb{E}(S_j^2) + \lambda_j \mathrm{Var}(S_j)\} \\ &\quad + \frac{1-\rho_i}{2}(1-\rho)\mathrm{Var}(D)(\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n))\frac{1}{n\mathbb{E}D} \end{aligned} \tag{3.2}$$

If we would now increase the total set-up time in a cycle $n\mathbb{E}D$ by a constant δ, while the variance of the total set-up time remains unchanged, equation 3.2 changes accordingly into

$$\begin{aligned} \mathbb{E}W_i &= \frac{1-\rho_i}{2}\frac{n\mathbb{E}D+\delta}{1-\rho} + \frac{1-\rho_i}{2}\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)\{\lambda_j\mathbb{E}(S_j^2) + \lambda_j \mathrm{Var}(S_j)\} \\ &\quad + \frac{1-\rho_i}{2}(1-\rho)\mathrm{Var}(D)(\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n))\frac{1}{n\mathbb{E}D+\delta} \end{aligned} \tag{3.3}$$

We see that the second term is independent of δ, while the first and third term are increasing and decreasing in δ respectively.

From equation 3.3 we can derive the optimal shift δ^* by equating the derivative of the right hand side with respect to δ to 0 as follows

$$\begin{aligned}
\frac{1-\rho_i}{2(1-\rho)} - \frac{\frac{1-\rho_i}{2}(1-\rho)\mathrm{Var}(D)\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)}{(n\mathbb{E}D+\delta)^2} &= 0 \\
\frac{(1-\rho)\mathrm{Var}(D)\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)}{(n\mathbb{E}D+\delta)^2} &= \frac{1}{1-\rho} \\
(1-\rho)^2\mathrm{Var}(D)\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n) &= (n\mathbb{E}D+\delta)^2 \\
\sqrt{(1-\rho)^2\mathrm{Var}(D)\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)} - n\mathbb{E}D &= \delta \qquad (3.4)
\end{aligned}$$

We ought to check if we have found a minimum here by looking at the second derivative of the right hand side of equation 3.3 with respect to δ. This second derivative reads

$$\frac{(1-\rho_i)(1-\rho)\mathrm{Var}(D)\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)}{(n\mathbb{E}D+\delta)^3}$$

Recalling that the h_{ij} are functions that can only yield positive values, we see that this expression is evidently positive for every queue i. Hence, from equation 3.4 we can derive the optimal shift

$$\delta^* = \Big(\sqrt{(1-\rho)^2\mathrm{Var}(D)\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)} - n\mathbb{E}D\Big)^+ \qquad (3.5)$$

Now, we see that the counter-intuitive effect that the mean waiting time of a certain queue i can be decreased by increasing the total set-up time in a cycle occurs only if

$$\mathrm{Var}(D) \geq \frac{n^2(\mathbb{E}D)^2}{(1-\rho)^2\sum_{j=1}^{n} h_{ij}(\rho_1,...,\rho_n)}$$

Analogously, this condition is necessary for the length of queue i to increase if the total set-up time in a cycle is cut. We conclude that to trigger the variance effect by altering the expected set-up time in a cycle in this case, the variance of the set-up times should be sufficiently large to begin with.

Chapter 4

Conclusion

In chapter 2, we considered an example of a particular polling system, in which queue lengths were growing without bound when set-up times are cut. This rather strange effect was found by Sarkar and Zangwill[10] and used an exhaustive service policy, a cyclical sequencing policy and a no-idling policy. This scenario containing the effect was used by Zangwill[13] to contest the Japanese production theory, therefore also called 'Zangwill's scenario'. The effect itself we called the effect the 'variance-effect'. Furthermore, we considered other possible service policies, sequencing policies and idling policies as well as the heuristical system policy by Van Oyen and Duenyas.
In chapter 3, we studied the behaviour of the queue lengths under each of the considered policies by examining one deviating policy while keeping all other policies and relevant variables the same as in Zangwill's example as much as possible. That way, we could deduce from the results what group of policies was responsible for observed changes in the behaviour of the queue lengths. Our results led to the following conclusions.

Group of policies	Effect
Service policies	While altering Zangwill's scenario by using different service policies, the behaviour of the queue lengths did not change substantially when set-up times were cut, apart from the sometimes observed initial 'stability-effect'. Therefore, the choice of service policy does not seem to influence the existence of the variance effect.
Sequencing policies	When using other sequencing policies than the one used in Zangwill's example, the behaviour of the queue lengths did not significantly deviate from the behaviour observed in case of the cyclical sequencing policy. Therefore, the choice of sequencing policy does not seem to influence the existence of the variance effect.
Idling policies	When Zangwill's scenario was considered, but idling policies were used based on which the server could opt for periods of idling, the behaviour of the queue lengths changed. No longer did they grow without bound as duration of set-up periods was cut. Hence, the choice of idling policy does influence the existence of the variance effect of the set-up time.

As a result of addressing the question whether the system policy used in Zangwill's scenario is the only policy in which the counter-intuitive effect shows itself, the answer found is simply no. All policies observed by us that did not allow the server to idle (cases in which the no-idling policy was used) showed the variance effect and allowed the queues to grow without bound. However, the choice of policies to be used is not totally irrelevant; we saw that in case idling policies are used such that the server can enter idling periods, the variance effect grows much feebler or is even non-existent.

We also considered the heuristic by Van Oyen and Duenyas, which was claimed to not show the variance effect either. We found that queue lengths could still grow in the least prioritized queues in certain circumstances, but as the set-up time kept decreasing, the length of all queues would decline eventually. When this heuristic is used, the server does assume idling periods, which may be the reason why the queues lengths did not grow unbounded.

However, based upon our findings we can assert that Zangwill's dismissal of the Japanese production theory should not be rejected based upon the policy used, as Duenyas[2] and Gerchak and Zhang[4] did. There certainly are policies used in practice which adopt the no-idling policy, for example due to technical constraints which render the server unable to do just nothing for an indefinite amount of time. Therefore, the variance effect is not merely theoretical. With that said, we do not want to intervene further in the discussion whether Zangwill's statement about the Japanese production theory is right or not. In this paper we focused on the system policy used, but there are so many more variables out there that should be examined carefully before judgement can be passed on the matter. We found for example that the distribution of the set-up period's duration must have a sufficiently large variance in order to even expose the variance effect.

As said before, we do now know that the idling policy does influence the degree of existence of the variance effect. We did not find any clues of the service policy and sequence policy influencing this existence.
Whereas we can state that the choice of idling policy is influential in this case, we only have a strong presumption that the choice of service and sequencing policies is not of influence. To be able to state this with complete certainty, we would have to consider each and every service and sequencing policy possible in each and every possible set of circumstances. By means of discrete time event simulation this is simply not feasible. This limits the strength of parts of our conclusion. Therefore, a suggestion for further research would be to create methods to consider these policies by means of analysis, although in literature it is shown that this is a rather hard task.

Bibliography

[1] Buyukkoc, C., Varaiya, P., Walrand J., (1985). The $c\mu$-rule revisited. (Advances in Applied Probability, Vol. 17, No. 2, pp. 237-238).

[2] Duenyas, I., (1994). The limitations of suboptimal policies. (Interfaces, Vol 24., No. 5, pp. 77-84).

[3] Fricker, C., Jaibi, R., (1994). Monotocity and stability of periodic polling models. (Queueing Systems, Vol. 15, pp. 211-238).

[4] Gerchak Y., Zhang, Z., (1992). The cheaper/faster-yet-more-expensive phenomenon: Are Zangwill's paradoxes indeed paradoxical? (Interfaces, Vol 24., No. 5, pp. 84-87).

[5] Levy, H., Moshe, S., Boxma, O., (1990). Dominance relations in polling systems. (Queueing Systems, Vol. 9, pp. 155-171).

[6] Liu, Z., Nain, P., Towsley, D., (1992). On optimal polling policies. (Queueing systems, Vol. 11, pp. 59-83).

[7] Oyen., M. Van, (1997). Monotonicity of optimal performance measures for polling systems. (Probability in the Engineering and Informational Sciences, Vol. 11, No. 2, pp. 219-228).

[8] Oyen, M. Van, Duenyas, I., (1995). Stochastic scheduling of parallel queues with set-up costs. (Queueing Systems, Vol. 19, No. 4, pp. 421-444).

[9] Sarkar, D., Zangwill, W.I., (1989). Expected waiting time for nonsymmetric cyclic queueing systems - Exact results and applications. (Management Science, Vol. 35, No. 12, pp. 1463-1474).

[10] Sarkar, D., Zangwill, W.I., (1991). Variance effects in cyclic production systems. (Management Science, Vol. 37, No. 4, pp. 444-453).

[11] Srinivasan, M.M., Gupta, D., (1996). When should a roving server be patient? (Management Science, Vol. 42, No. 3, pp. 437-451).

[12] Winands, E.M.M., (2007). Polling, production & priorities. (PhD Thesis, Eindhoven University of Technology).

[13] Zangwill, W.I., (1992a). The limits of Japanese Production Theory. (Interfaces, Vol. 22, No. 5, pp. 14-25).

[14] Zangwill, W.I., (1992b). Response to comments on our work by Duenyas, by Gerchak and Zhang, and by McIntyre. (Interfaces, Vol. 24, No. 5, pp. 90-94).

Appendix A

The set-up time distribution

For the set up time distribution a convenient distribution is used consistently throughout the several simulation scenarios.

Assume that X is a random variable distributed according to this distribution. Then this distribution, also used by Zangwill[13] in a slightly modified form, has the pleasant property that with the change of one single parameter $\mathbb{E}X$ can go down rapidly, while the term $\frac{\mathrm{Var}(X)}{\mathbb{E}X}$ goes up simultaneously. This is very useful since exactly those characteristics would invoke the anomalous effects described in section 2.2.

The probability distribution of X is defined as follows:

$$X = \begin{cases} \frac{2}{u^2} & \text{with probability } 1 - \frac{1}{u^3} \\ 2u & \text{with probability } \frac{1}{u^3} \end{cases}$$

Here,

$$\mathbb{E}X = \frac{2}{u^2}(1 - \frac{1}{u^3}) + 2u(\frac{1}{u^3}) = 2(\frac{2}{u^2} - \frac{1}{u^5})$$

and

$$\mathbb{E}X^2 = (\frac{2}{u^2})^2(1 - \frac{1}{u^3}) + (2u)^2(\frac{1}{u^3}) = 4(\frac{1}{u} + \frac{1}{u^4} - \frac{1}{u^7})$$

Hence,

$$\mathrm{Var}(X) = \mathbb{E}X^2 - (\mathbb{E}X)^2 = 4(\frac{1}{u} + \frac{1}{u^4} - \frac{1}{u^7}) - 4(\frac{2}{u^2} - \frac{1}{u^5})^2 = 4(\frac{1}{u} - \frac{3}{u^4} + \frac{3}{u^7} - \frac{1}{u^{10}})$$

.

Now, it is easy to see that when u increases, $\mathbb{E}X$ decreases considerably. However, $\frac{\mathrm{Var}(X)}{\mathbb{E}X}$ grows without bound when u keeps increasing. To see this, let us consider the case where u approaches infinity.

$$\lim_{u\to\infty} \frac{\mathrm{Var}(X)}{\mathbb{E}X} = \frac{4(\frac{1}{u} - \frac{3}{u^4} + \frac{3}{u^7} - \frac{1}{u^{10}})}{2(\frac{2}{u^2} - \frac{1}{u^5})} = \lim_{u\to\infty} \frac{2u^9 - 6u^6 + 6u^3 - 2}{2u^8 - u^5} =$$

$$\lim_{u\to\infty} \frac{2u - \frac{6}{u^2} + \frac{6}{u^5} - \frac{2}{u^8}}{2 - \frac{1}{u^3}} = \lim_{u\to\infty} \frac{2u}{2} = \infty$$

Indeed, $\frac{\mathrm{Var}(X)}{\mathbb{E}X}$ diverges to infinity, so the conditions needed to provoke the anomalous effect described in section 2.2 are met.