Modeling operators in a queueing network

Mark van 't Klooster

Masters Thesis

Stochastic Operations Research Department
Faculty of Applied Mathematics, University of Twente

December 2003

Committee:

prof. dr. W.H.M. Zijm
dr. J.C.W. van Ommeren
ir. M.F. van Assen, Add-Smart BV
ir. R.J. Mantel, Faculty of Mechanical Engineering

Contents

Preface

The aim of this thesis was to develop a model of the effect of operators on the sojourn times in a manufacturing system, especially in a *Multi-Class Generalized Kanban Controlled System* (MC GKCS) by extending a current model. It was carried out as the final project of my masters program in applied mathematics, and done with Add-Smart BV in Almelo.

The old model for a MC GKCS was already implemented into a software package. The second part of the project was to implement the new model in this software.
When looking at the implementation of the actual algorithm, it became clear that it was not easy to implement new aspects in the programming code. This code was absolutely not flexible. In the current implementation, this code was already treated as a black box, and thus it was not guaranteed bug-free. This also became clear in using it, the black box was not tested well enough. In some situations it returned a *division by zero*-error, which is not allowed to happen in well-tested code.
Because of all this problems, we have decided that it was best to do the implementation again. Then it could be fit in more in the used object structure, and the total program code could be optimized. This was not possible in the current version, because the black box needed very specific data, which is not useful for more efficient implementations. But this change would also affect the Graphical User Interface, therefore it was a lot of work, but it was the only option. Because of this amount of work, there was no time to perform exhaustive tests with the new model. For large models, the results of the algorithm can only be validated with simulation results. Because these were not available in short time, the algorithm is tested only for very small networks.

I would like to thank all the people at Add-Smart BV for providing a nice place to work and support with all Delphi problems during the first part of the project. I would like to thank all the people from the SOR-group for providing a place to work (very close to the coffee machine) during the second part of the project and for the nice interruptions during coffee moments. Especially I would like to thank Jan Kees van Ommeren who was my supervisor. Furthermore I would like to thank Ellen for her moral support during the project, and Ruud for his support during the last few weeks. Finally, I would like to thank everybody of TW and Abacus for the nice time during this project.

Summary

In many manufacturing systems, products do not only need a machine for their different operations, but they also need the care of an operator during (the beginning of) the service time of each operation. Because these operators are not always available, they form a restriction for the capacity and thus for the performance of a manufacturing system. In this thesis, an iterative method is developed to analyze manufacturing systems with operators, especially for manufacturing systems operating under a kanban policy, a *Multi-Class Generalized Kanban Controlled System*.

Chapter 1

Introduction

1.1 Modeling manufacturing systems

Numerous studies have been conducted on the modeling of manufacturing systems with the help of queueing networks. In 1983, Whitt [Whi83] presented the *Queueing Network Analyzer (QNA)*, which provided an approximate method for general open networks. Since then also other commercial PC-based software packages have been released such as *ManuPlan* [SDD86] and *MPX* [SdT91]. Such software packages have facilitated the step of queueing network analysis out of the academic (= research) environment and into the mainstream of manufacturing systems analysis. However, the disadvantage of systems like MPX is that only single stage open systems can be analyzed. These systems are push-systems, in which it is difficult to control the workload. For this workload control, pull systems are better because the maximum number of products in the system can be set.

Buzacott and Shanthikumar [BS93] introduced a system for coordinating and controlling the material and part flow within a multi-stage system, called the PAC (Production Authorization Card) system. It was then demonstrated how, through the appropriate choice of parameters, the PAC system can be specialized into a wide variety of classical coordination approaches. The PAC system is a general system that includes batching of parts and time-delays to deal with advanced information on the demand. A PAC system with unit batch sizes and zero time-delays reduces to the Generalized Kanban Control System (GKCS). As with the classic kanban system a workload control rule is imposed on each manufacturing stage by limiting the amount of work-in-process to a certain number of kanbans, i.e. a work-package is only allowed to enter a group when a kanban is available. At the end of each manufacturing stage an order-up-to, or base stock level can be defined for the output buffer of that stage. The system will then try to reach that level of stock.

There has been a variety of attempts to analyze multi-stage single class kanban systems using stochastic models. Numerous papers report on kanban based systems with stages in series using a discrete time stochastic model, which would be appropriate when kanbans are collected, and buffers are checked at periodic time intervals; see for instance [DHMO89, GY94, PC99].
Other models, however, use a continuous time control mechanism. The behavior of the system is then described by a continuous time Markov chain (CTMC) from which exact or approximate solutions can be obtained. Karmarkar and Kekre [KK89] study single and two

stage kanban systems where the CTMC's are solved using numerical techniques.
Legato et al. [LBR89] analyze a kanban system with a number of Z stages in series. Each stage consists of a single station with exponential service times and the system is saturated with demands. The CTMC is analyzed using numerical techniques. The same system is studied by Mitra and Mitrani [MM90], but they use an approximate method where the system is decomposed into its stages. Each stage is then solved exactly and an iterative algorithm is used to determine the unknown parameters of the stage. Mitra and Mitrani [MM91] have extended this method to the case of stochastic external demand.
Di Mascolo et al. [MFBD93] view a GKCS as a multi-class closed queueing network by considering the Kanbans of each stage as one class of customers. Subsequently, a product-form approximation technique is used to solve the obtained CQN. For a single-class classic Kanban system with stages in series, Di Mascolo et al. [MFD96] used an extension of the method of Marie [Mar79] to analyze each stage in isolation where complex manufacturing cells are allowed in each stage.
More recently, Baynat et al. [BDMF01] presented a new way of deriving the analytical method presented in [MFD96]. Their new approach is to view the queueing network of the kanban control system as a multi-class queuing network in which each kanban loop is represented by a class of customers. This allows them to use the general technique proposed in [BD96] for analyzing multi-class queuing network using product-form approximation methods. In terms of equations, the new method is equivalent to the one presented in [MFD96]. However, the computational algorithm is much more efficient since it avoids the two levels of iterations involved in the original algorithm. Another major advantage of the new method over that originally proposed in [MFD96] is that it provides a general framework for the analysis of more general kanban systems. Indeed, the authors show in their paper how the approach can easily be extended in order to handle kanban systems with multiple consumers and multiple suppliers, kanban controlled assembly systems and generalized kanban systems.

Only a few attempts have been made to analyze a kanban system with multiple part types. So [So89] used a bulk queue model to obtain the average proportion of backordered demands, which can then be used to approximate the number of kanbans needed to attain a certain performance level. This model, however, is limited to a single exponential server per stage.
A method that is able to handle multiple classes is developed by Wormgoor [Wor01] and follows the same lines of the method of [MFD96]. Again the system is decomposed into its stages. The key to the analysis is the fact that each stage can be seen as an Open Queueing Network with Population Constraints (OQN-PC), i.e. a network with a workload control rule (the population constraint) where production requests and finished parts of the previous stage wait in an external queue until they are allowed to enter the network. These OQNs-PC can be analyzed as a closed queueing network with flow equivalent servers, see e.g. [Dal90, BS93, Bui98]. Extensions to multi-class systems, i.e. where different part types are produced in the same resource group, are presented by Baynat and Dallery [BD96], and Buitenhek et al. [BvHZ99, BvHZ00]. The latter distinguishes between general and class-dependent workload norms. Based on the work of Buitenhek [Bui98], Wormgoor [Wor01] uses an iterative algorithm to obtain the unknown parameters of each stage.

Van 't Klooster [Klo03] developed a prototype Decision Support System (DSS), called SmartSCManager, which overcomes the disadvantages of the existing software packages by using a model of a Multi-Class Generalized Kanban Controlled System (MC GKCS).

This DSS helps decision makers to make good decisions about the design of a manufacturing system. For the analysis of this model, he has adopted the algorithms of Wormgoor [Wor01], because they seem to be the most general.

1.2 Multi-Class Generalized Kanban Controlled System

A Generalized Kanban Controlled System is basically a base stock system with a workload control rule for each stage. The workload control rule is imposed on each stage by limiting the number of products in each stage by a fixed number of kanbans. A production request can only be taken into production when a kanban is available.

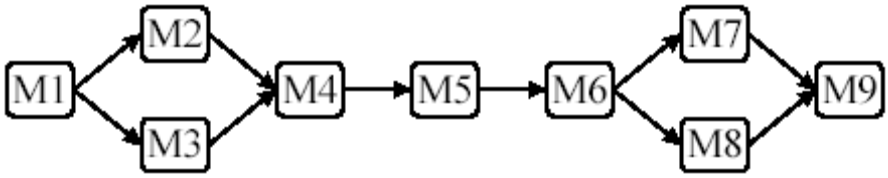

Figure 1.1: Example of a manufacturing system.

The GKCS will be described by means of an example. Consider a manufacturing system with nine stations as in figure 1.1, which produces a single part type. The manufacturing system is divided into $Z = 3$ stages. Each stage-i consists of a *production process*, an *output buffer* for finished parts and a *kanban box* where kanbans not currently in production are stored, see figure 1.2. Each stage-i, $i = 1, ..., Z$, has a fixed number of kanbans, N_i. Note that when $N_i = \infty,\ i = 1, ..., Z$ the GKCS reduces to the classic Base Stock system, since the workload control rule is released. To explain the system control let us first follow the flow of parts and kanbans in stage-i.

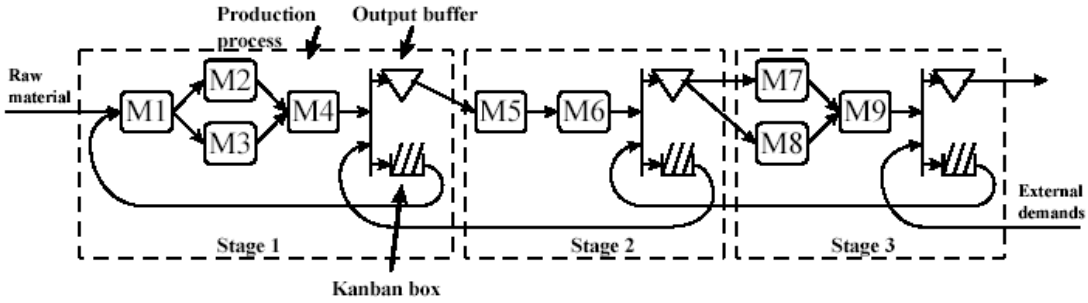

Figure 1.2: Example of a 3 stage Generalized Kanban Controlled System.

When a demand for a stage-i finished part arrives, represented by a stage-$(i + 1)$ kanban or an external demand, and there is a finished part available, the kanban is attached to the part and enters the stage-$(i + 1)$ production process or the part is sent to the customer. When no finished parts are available, the demand is *backordered* until a stage-i finished part becomes available. On arrival of a demand at stage-i a *production request* for a stage-i part is generated to replenish the (future) consumption of the stage-i finished part by the demand. When there are no stage-i kanbans available in the kanban box, the production

requests are backordered. When a stage-i kanban is available, the production request is attached to the kanban. Together they travel upstream and form a demand for a stage-$(i-1)$ finished part. When a stage-$(i-1)$ part is available the kanban and production request are attached to the part which then is taken into production. The part is then processed in the production process according to its routing. Upon completion of the production process at stage-i the part travels to the output buffer and the kanban is detached and is placed in the kanban box. The production request is rendered obsolete since this concludes the replenishment cycle of a stage-i part.

Next consider the arrival of an external demand at the manufacturing system in figure 1.2. The parts that are in the output buffer of stage-3 are the finished products of the manufacturing system and are used to fulfill the external demand. When the demand cannot be met immediately (i.e. the buffer is empty) the demand is backordered until a stage-3 part becomes available. As mentioned above, upon the arrival of an external demand a production request for a stage-3 part is generated. When a stage-3 kanban is available, it travels upstream and becomes a demand for a stage-2 finished part. Consequently a production request for a stage-2 part is generated. In the same way a demand and a request for a stage-1 part will be generated. We assume that there is a sufficient supply of raw material for stage-1. Thus when a stage-1 kanban is available to match with a stage-1 production request it can be taken into production immediately. In this way information of an external demand is transferred upstream from a downstream stage as long as there are kanbans available to match with a production request.
The behavior of a GKCS also depends on the *base stock level* of each stage, denoted by S_i, $i = 1, ..., Z$. The base stock level is based on the start-up conditions of the system, that is when all kanbans are in the kanban box. The number of parts present in the output buffer at that moment is the base stock level. Since a production request is generated for each demand of a part in an output buffer to replenish the (future) consumption of that part the system always tries to have S_i parts in the output buffer of stage-i.

A special case of a GKCS is when $S_i = N_i$, for $i = 1, ..., Z$. Then a GKCS reduces to the classic kanban system. The advantage of a GKCS over a classic kanban system is that information about demands can be faster transferred upstream. Consider the arrival of a demand at stage i when the output buffer is just empty, i.e. there are no backordered demands. In the case of the classic kanban system the production request would also have to wait since all kanbans would be in production (or waiting for a stage-$i-1$ finished part). However if the number of kanbans is larger than the base stock level the production request can already be transferred upstream. Thus, when the number of kanbans exceeds the base stock level, information is transferred upstream faster.
The GKCS can be modeled as a *queueing network with synchronization mechanisms*. For a multi-class GKCS with R job classes and Z stages, each job class has an individual base stock level and workload control rule, i.e. number of kanbans, in each stage. Thus each job class has its own synchronization stations, and the synchronization stations are single class stations. The queueing network model of a multi-class GKCS is given in figure 1.3. To each variable an index r, $r = 1, ..., R$ is added, denoting its job class.
To analyze this queueing model, it is decomposed into its stages. The stages, with the synchronization stations, can be analyzed in isolation if the external arrival rates are known. From this analysis, the service rates of all synchronization stations of the stage are determined, and these new values are used in the analysis of the next stage. When the last stage

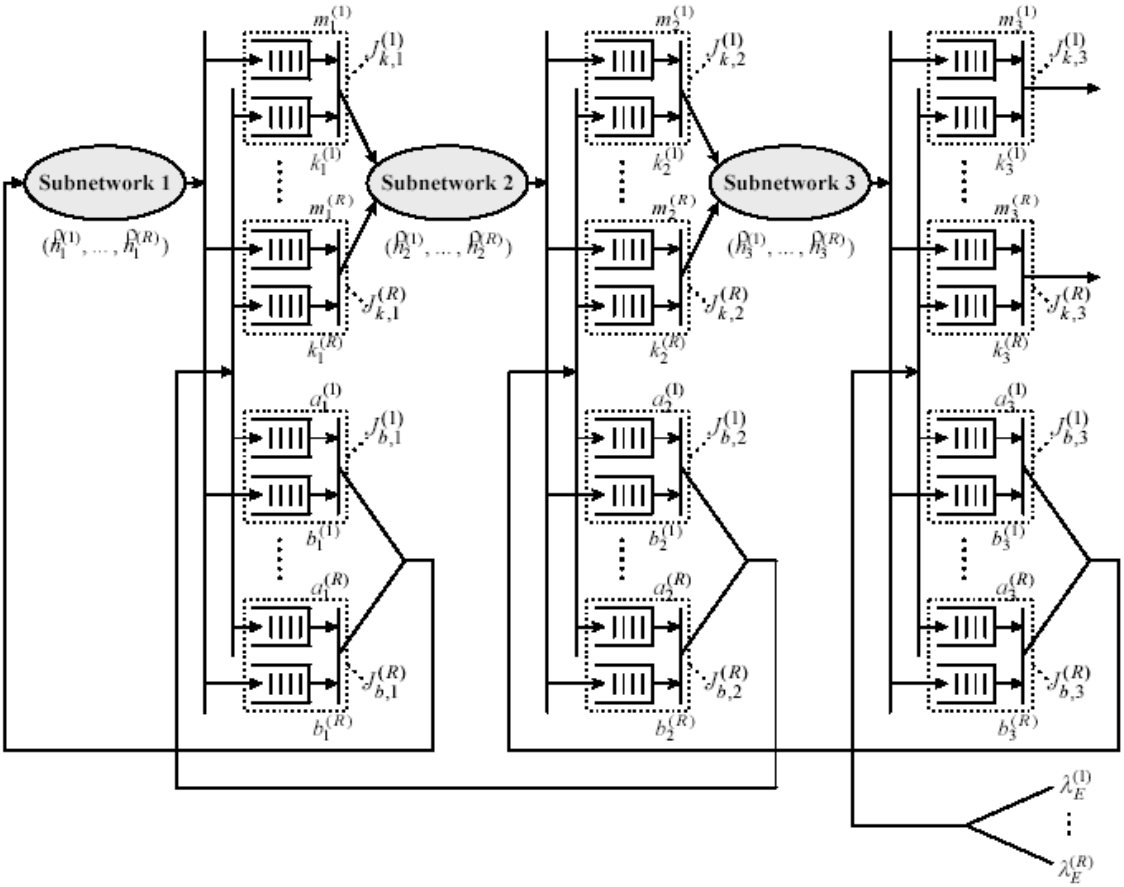

Figure 1.3: Queueing network model of a multi-class GKCS.

is analyzed, this process is continued backwards until the first stage is analyzed again. This process is repeated until convergence of the service rates of all synchronization stations of all stages.
To obtain the service rates of the synchronization stations in one stage, also an iterative algorithm is used. The service rates of one synchronization station can be obtained by analyzing the complement network of this synchronization station. This complement network is the whole network of the stage, without the synchronization station. It is analyzed with an AMVA-algortihm. This process is repeated for every synchronization station in the stage until convergence of all service rates. Both this iterations are given in appendix B.

From the analysis of the last synchronization station $J_{k,Z}$ of every product, the market service levels can be obtained. These market service levels are a measure for the general performance of the manufacturing system. For every product type these market service levels are:

Fill rate
Percentage of demands that can be fulfilled without delay.

Number of finished products
Number of products that leave the system.

Stockout probability
Percentage of time that there is at least one backordered demand waiting to be fulfilled.

Number of backordered demands
Number of demands that could not be fulfilled directly.

Customer order response time
Waiting time before a customer order is fulfilled.

Waiting time of a backordered demand
Waiting time before a backordered demand is fulfilled.

These market service levels can be computed with the equations given in appendix C.

1.3 Outline of this thesis

Van 't Klooster [Klo03] proposed some aspects that will make it possible to upgrade the SmartSCManager to a better and more useful system. Most of these proposals are just ways to make the Graphical User Interface better understandable for the user. However, in many manufacturing systems, products do not only need a machine for their different operations, but they also need the care of an operator during (a part of) the service time of each operation. Because these operators are not always available, they form a restriction for the capacity and thus for the performance of a manufacturing system. In this thesis, a method is developed to analyze manufacturing systems with operators, especially for manufacturing systems operating under a kanban policy.

The outline of this thesis is as follows. Chapter 2 will give a formal formulation of the problem. Also existing models are given. Because the existing models do not solve all problems, a new algorithm for analyzing a manufacturing system with operators is given in chapter 3, first for a simple network, and extended to a Multi-Class Generalized Kanban Controlled System. The algorithm developed in chapter 3 can be tested and compared with analytical results, which is done in chapter 4 for some simple cases.

Chapter 2

Operators

Suppose a factory where two products, A and B, are manufactured on two machines. Product A needs machine 1 and product B needs machine 2. The products arrive at their machine according to some process, which is assumed to be stochastic. Before an operation on a machine can start, it first needs the attention of a operator, who has to setup the product to the machine. When a product arrives at a machine, two situations can occur. First, it is possible that the machine is free, but that the operator is busy on the other machine. In that case, the product has to wait longer before it can be operated. Second, it is also possible that the operator is finished at a machine, and that no product can start on the other machine. Thus the operator can be idle.
This is a small example of the problem described in this thesis. This chapter gives the general formulation of the problem and an overview of existing models describing such a system in the literature.

2.1 Problem formulation

A manufacturing cell is assumed to consist of m machine groups. Every machine group contains c_m identical machines. These machine groups are used to produce r different products. Each product requires operations on several machine groups, according to a (stochastic) route over these machine groups. The service time of these operations is generally distributed, described by the mean rate and the squared coefficient of variation (SCV). During every operation, the attention of an operator is needed. This can be for the whole operation time, or only during the setup time, for example to set a product on a machine. There are p operator groups, each containing several identical operators.
The question is what the performance of this system is, and especially what the effect is of the waiting time of a product at an operator group.

2.2 Literature overview

Because of the complexity and the denial of the importance of this problem, only a few researchers have looked at this problem. However, for some specific situations an approximation exists.
A model that can be used for this problem also is the model for *machine interference*. This problem originates from the problem of several machines with breakdowns were one or more repairmen are available to repair these broken machines. The operator problem can

be modeled as a special case of the machine interference problem sketched above.
The classical machine interference problem considers a set of machines that need a repairman if they break down. Machines can be in two states, up or down. When it goes down, it waits for a repairman to be available. The result of the model is the availability of a machine, which can be used to determine the effective service time on that machine.
The machine-operator problem and the machine interference problem are identical:

1. A machine is *up* if it is idle in the machine-operator problem.
2. A machine is *down* if it is occupied by a customer.
3. The waiting time for an available repairman is the waiting time for an operator. With this waiting time, the effective service time of an operation can be set.

In computer science, there is a similar problem where a process needs multiple processors or other hardware to be completed. Due to technical constraints this can cause waiting times, for instance if a hard-disk cannot deliver information to a processor at the same speed the processor processes the data. This problem is know as *simultaneous resource possession*. In special situations this can lead to a problem with the same characteristics as the operator problem. Further, some very different models exists, which will be discussed at the end of this section.

2.2.1 Machine interference

Stecke and Anderson [SA85] give a review of models for the machine interference problem that have been developed up to 1985. Sztrik [Szt01] provides a list of articles with theoretical and practical developments up to 2001. This recent developments mainly focuses on extensions which are not used in modeling the machine-operator problem, like the availability of spare parts or machines that cannot be repaired.
The classical machine interference problem assumes exponentially distributed processing times and operator service times. This can be modeled as a finite source multiple server exponential queueing system, where the operators are the servers and the machines are the customers. Since the number of machines (K) is finite this can be modeled as a $M/M/m/K$ queueing system. Sztrik and Bundy [SB93] analyze this with one repairman. Ching [Chi01] considers multiple operators and a repair process being l-phase Erlang distributed. Chakravarthy and Agarwal [CA03] analyze a model with multiple machines, a single operator, exponential distributed arrival times and phase type service times and repair times. Eben-Chaime [Ebe98] developed a simple recursion for the model with a single product, multiple machines and repairmen, exponentially distributed arrival and service times and general distributed repair times.

2.2.2 Simultaneous resource possession

Kurasugi and Kino [KK99] use a two-layer queueing model to approximate the performance measures of queueing models with *simultaneous resource possession*. They consider a queueing network consisting an upper and a lower layer. The upper layer consists of a number of stations and the lower layer consists of a number of nodes. Every station and node have several servers. When a customer arrives at a station, it waits for an available server at that station. When a server becomes available, the customer is routed directly to the lower layer. In the lower layer, the customer will travel through the nodes as routed by a routing matrix

associated with a station. When the customer is finished in the lower layer, it is routed to the upper layer according to a routing matrix. The service discipline at a station is FIFO (First In First Out), the service discipline at a node is assumed to be FIFO, PS (Processor Sharing) or LIFO-PR (Last In First Out with Preemptive Resume). The service time at a node for customers of a station is assumed to be exponentially distributed. Further, it is assumed that there are no limitations on the queue lengths in the network.
First, Kurasugi and Kino [KK99] show that the equilibrium distribution of the occupancy has a product form if the capacity (number of servers) for any station in the upper level is not limited. However, this is not the case if the capacity is limited. To approximate the performance measures in this case, they split the two layers. Using the lower layer they approximate the service rate at station r as

$$\mu_r^*(\mathbf{n}) = v_r \frac{G(\mathbf{y} - e_r)}{G(\mathbf{y})} \tag{2.1}$$

where $\mathbf{n}$ the number of customers in the upper layer, $\mathbf{y}$ the number of customers traveling through the lower layer, v_r the relative frequency at which a customer visits station r and $G(\mathbf{n})$ the normalization constant which can be calculated using standard techniques. With these service rates the equilibrium distribution of $\mathbf{n}$ can be approximated, and thus the performance measures.
For this equilibrium distribution, they use a product form approximation

$$P(\mathbf{n}) = \Phi(\mathbf{n})\mathbf{v}^{\mathbf{n}} \tag{2.2}$$

where $\mathbf{v}^{\mathbf{n}}$ is defined as $v_1^{n_1} v_2^{n_2} \dots v_k^{n_k}$ and $\Phi(\mathbf{n})$ is determined by the recursion

$$\Phi(\mathbf{m}) = \frac{\sum_{i=1}^{R} \Phi(\mathbf{m} - e_i)}{\sum_{i=1}^{R} \mu_i^*(\mathbf{m})} \qquad \forall \mathbf{m} \leq \mathbf{n} \tag{2.3}$$

With this model, the machine-operator problem can be seen as follows:

1. The upper layer is the machine network.
2. The lower layer is the operator queue.

The waiting time for an operator of a product on machine m can be computed from the service times of station m approximated with (2.1) by subtracting the given service times of the operation.

Wormgoor [Wor01] proposes the same splitting method as [KK99] but with the lower layer modeled as a *Semi Open Queueing Network (SOQN)* as proposed in [Bui98]. A SOQN is a queueing network with a fixed number of pallets circling through the inner network. Customers arrive at the the network and wait for a pallet to be available. With this pallet, they travel through the inner network until the customer is ready to leave the network again. The pallet then picks up a new customer, if one is waiting. This SOQN is modeled in detail in [Bui98] for the most general situation. Buitenhek models the SOQN as a closed queueing network and gives an aggregation method to finding approximate solutions.
Another approximation is given in [SH02], which is based on the work of Buitenhek. For the case with a single product, they give an approximation for the inner network with one or two stations. With more servers they approximate the inner network by two stations,

one being the bottleneck station, and one being the aggregation of the other stations. This method is extended for multiple products in [CHSZ02].
Wormgoor takes an inner network consisting of two stations, one with an ample server (server with no waiting time, thus an infinite number of servers) for products that do not need an operator, and one server representing the operators. Products can have a arbitrary routing through this network. The number of pallets circling around equals the total number of machines, which is the maximum number of jobs that will enter the network. This inner network is a generalization of the one given in [Bui95], where it is assumed that the operation always needs an operator during its whole service time.
The cycle time through this SOQN contains the waiting time for an operator and can be used to set new service times. With these new service times, the whole manufacturing network can be analyzed. Because nothing can be said about the arrival process at a single machine, Wormgoor proposes to approximate this with a Poisson distribution.
Desruelle and Streudel [DS96] consider a system with identical stations and several products that need service on every machine in line. They assume one operator needed for the loading and unloading a product on a machine. They also split the system into the same two networks as above, the part-machine network (upper layer) and a machine-operator network (lower layer). They model the part-machine network as an open queueing network, and the machine-operator network as a closed queueing network. To analyze the first they use a MVA-technique. For the machine-operator network, they develop an iterative method to find the coefficient of variation of the interdeparture time of this network. Then the well known approximation for the waiting time in a $G/G/1$ queue is used to find the waiting time for an operator. The results of this method are good if the utilization is not too high.
Adan and others [AVW98] uses *Multi Server and Concurrent Classes of Customers (MSCCC) queues* as a special class of queueing models with simultaneous resource possession. This model uses multiple customer classes, multiple machines and an operator pool with several operators. Customers arrive according to a Poisson process and are served with FCFS discipline and the serving times are exponentially distributed with different rates for different customer classes. It is know that the equilibrium distribution of an MSCCC queue has a product form if the service times are independent of the job type.
For job dependent service times, two special cases are analyzed: only one operator and as many operators as machines. They use this property to find this distribution by dividing the state space into two sets, the interior equations and the boundary equations, and substitute a product form distribution in these equations. Then these equations can be solved to obtain the equilibrium distribution. For models with less operators as machines, the solution will not be a product form. This makes them harder to analyze. By looking at the solutions of the two special models, [AVW98] gives a suggestion for further research.

2.2.3 Other models

A very different approach is the method of Gold [Gol01]. He considers r products arriving at m machines and p operators. All the arrival and service rates are assumed to be generally distributed. He aggregates all machines and all the operators to one machine and one operator. The arrival and service rates are just the sum of all the different arrival and service rates. In this new model, when a customer arrives, he joins two queues, one for the machine and one for the operator. Let X be the number of customers in the machine queue and Y the number of customers in the operator queue. Under the assumption that the load of the machine and the operator are equal (ρ), Gold derives the following expression for the

expected maximum of X and Y:

$$E[max(X,Y)] = \frac{1}{\rho - 1}\left(\frac{3}{2} - \frac{1}{8\rho}\right) \tag{2.4}$$

With this he derived the *Flow factor* (FF), which is the factor with which the service time increases:

$$FF = c^2\left(\frac{1}{2\rho} + \frac{7}{8}\right)\frac{\rho}{1-\rho} + 1 \tag{2.5}$$

where c^2 is the variability of the aggregated system. Unfortunally he does not say how to approximate this variability. The comparison with simulation results shows that under medium and high load, the results are too optimistic if the squared coëfficient of variation of the service time is > 1 and too pessimistic if the squared coëfficient of variation of the service time is < 1.
Gold also states that experience shows that the factor with which the lead time increases is between 1.3 and 1.5 if operators are incorporated, depending on the load. This is in line with the above Flow Factor.

Mittler and Kern [MK97] model the problem as an closed polling model and use a discrete time approximation to find the performance measures. In a polling model, a server cycles through several queues and serves all customers in that queue. When the queue is empty, the server moves to the next queue on the cycle. Mittler and Kern consider a model with multiple machines with one server, one repairman and generally distributed failure times and repair times. The repairman serves the machines in a cyclic manner. Every machine contains one customer. After being served, the customer is routed directly back to his machine and wait again for service. The repairman moves to the next machine in the cycle. The walking times of the repairmen are considered to be generally distributed. The result is an iterative algorithm to compute the steady-state probability of the number of jobs in the system and the steady-state probability of the waiting time distribution for a customer. This algorithm works well for large symmetric systems. When a system is asymmetric, the computational complexity increases.

2.3 Conclusion

The modeling of operators does not get very much attention in the literature. Only a few articles exist in which this problem is analyzed. None of the methods found works for a general manufacturing system. They all have restrictions which make them not very useful in practice. These models have the same basic ideas:

1. Assumption: Jobs (customers) first arrive at a machine. When a machine is available, the job occupies this machine and together they wait for an operator to become available.

2. The waiting time for an operator is approximated by solving a queueing system.

3. This waiting time is used to set new service times on a machine.

4. With this new service times, the performance measures of the whole manufacturing system can be approximated.

The difference between the existing models is the method for approximating the performance measures of the two queueing systems given the structure of a specific case. For different queueing systems many approximations exist.

For practical situations the basic idea is not useful, because of complexity issues. Thus another approximation has to be found. In the next chapter, such an approximation is developed, which does not use the ideas given above.

Chapter 3

Modeling operators

The basics found in the previous chapter for the analysis of the operator problem seems to give good approximations. But for practical problems (with a large number of products and machines) it is not a good option, because two or more networks have to be analyzed, in stead of one. For a MC GKCS, this means that the steps in algorithm B.2 are doubled. This is done for every iteration step in algorithm B.1, thus the calculation time will increase severely. And even without operators, analyzing a MC GKCS already has a large calculation time. To try to solve this problem, another method to approximate the influence of the operators is presented, which is applicable in larger networks.

This chapter starts with the analysis of the problem for an *Open Queueing Network* in section 3.1. After this, a new iterative algorithm for this network is presented in section 3.2.1. In section 3.3, this algorithm is extended to a MC GKCS. Finally, some remarks about the new algorithm are given in section 3.4.

3.1 Open Queueing Network with operators

First consider an *Open Queueing Network* (OQN). This is the most simple queueing network, and therefore used to start the analysis. In the analysis, the following assumptions are done:

- An arriving product first occupies a machine server, and together they wait for an operator. This because an operator will be more expensive then a machine server. Therefore an operator must be able to work as efficiently as possible.
- There is only one operator group with p identical operators.

With the first assumption, the machines change to synchronization stations. The arrival of products at the machines consists of two streams, *external arrival* and *internal arrival*. External arrival is the arrival of products from outside the network. Internal arrival is the arrival of products from other machines in the network.
When a product is waiting at a machine and a machine server is free, they move together to the queue for the operator group. When an operator is free, the product is served. After being served by the operator, it moves, together with the machine server, to an ample server. This is a server without a queue, thus arrival products are served directly without waiting. The service time here is the part of the total service time where no operator attention is

needed.
When the total service of a product on a machine is finished, the machine server is routed back to the synchronization station of that machine, and the product is routed to its next machine where it has to be served according to its route, or it leaves the system if it is finished. Figure 3.1 shows a diagram of this process.

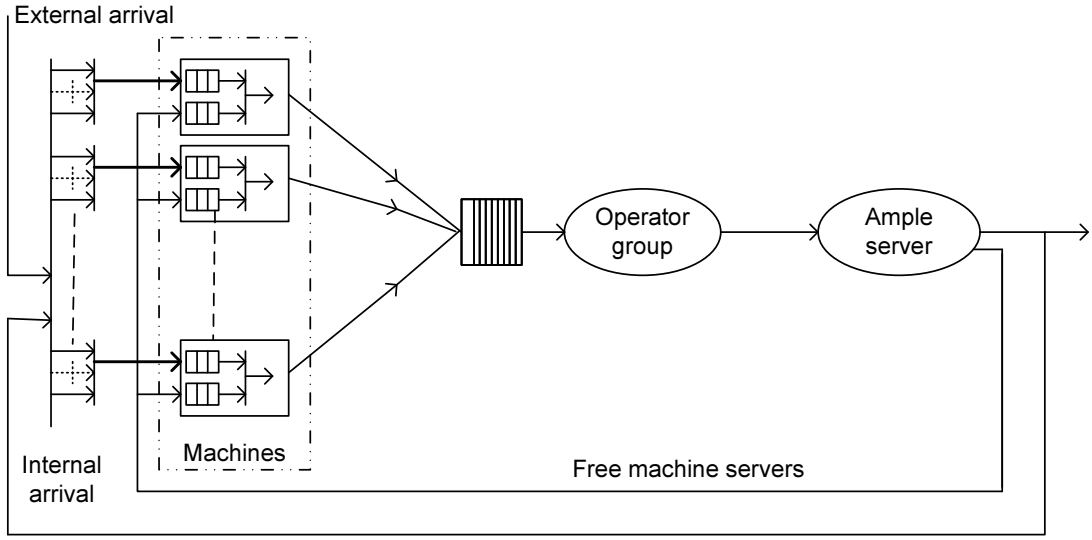

Figure 3.1: Queueing network with operator group

When looking at figure 3.1, we notice that, without the routing back of the products, the network is a *multi-class semi open queueing network* with pallets dedicated to a subset of the products, and with two stations. With the routing of the products, the network can be seen as a *multi-class classic kanban system* with one cell and stochastic routing through this cell.

For multi-class classic kanban systems, no solution exists in literature. The MC GKCS-algorithm, given in section 1.2, can be used, with $N = S$. But this assumes exponential arrivals to the synchronization stations, which will not be the case in general. But using this model as an approximation does not solve the problem of time complexity for larger networks.
For semi open queueing networks, solutions exist for multi-class networks [Bui98, SH02, CHSZ02]. Srinivasan and Heragu [SH02] approximate all interarrival processes in the network and used the obtained SCV's of these. With these processes, the average queue length can be determined.

Inspired by the work of Srinivasan and Heragu [SH02], we suggest to approximate all the departure processes, and thus all arrival processes, in the network. When these processes are known, performance measures can be estimated. This means that for every queue in the network the departure process have to be obtained. To do so, start by setting the arrival of the first machines to the external demand. Then the departure processes of these machines can be computed and used as the arrival process to the operator queue. The departure process of the operator queue is the arrival of the ample server. Together with the routing

through the machines, the departure of the ample server resets the arrival processes at the machines. This process can be repeated until the departure processes, and thus the arrival processes at the machines, do not change anymore. With all this arrival processes, the average waiting time in every queue can be obtained, and thus the expected sojourn time in the network.

The rest of this section will give a more detailed description of this iteration, and how to obtain the different processes and performance measures. But first, for the sake of completeness, some important parameters of a queue are given.

3.1.1 Queue basics

Three stochastic processes are used for the analysis of a queue: the arrival process, the service process and the departure process. These stochastic processes are characterized by a mean rate (μ) and a squared coefficient of variation (SCV). This SCV (C^2) is defined as

$$C^2 = \frac{\sigma^2}{(ES)^2} \tag{3.1}$$

where $ES = \frac{1}{\mu}$ and σ^2 the variance.

The *utilization* ρ of a queue is defined as

$$\rho = \mu_a ES_s \tag{3.2}$$

where μ_a is the mean arrival rate and ES_s the mean service time.

The queue is called *stable* if $\rho < 1$. In the rest of the analysis, every queue is assumed to be stable.

The queue contains a number of servers c.

3.1.2 Departure process

An important question is how to obtain the departure process of a queue. This departure process can be approximated from the arrival process and the service process.
The mean departure rate of the queue is easy to obtain. If the queue is stable, this is equal to the mean arrival rate, because what goes in, must go out.
The SCV's are more complex to obtain. Several researchers have given approximations for the SCV of the departure process. The most commonly used is the one given by Whitt [Whi83]:

$$C^D = 1 + \frac{\rho^2}{\sqrt{c}}(C^S - 1) + (1 - \rho^2)(C^A - 1) \tag{3.3}$$

C^D, C^S and C^A are the SCV of respectively the departure, service and arrival processes.

This approximation for the departure process assumes a single arrival and single service process. For the network in figure 3.1, this will not be the case, in general. There will be multiple arrival processes with different service processes. This can be solved by aggregating these processes into a single process. Then, equation (3.3) can be used to approximate the SCV of the departure process. After this, the departure process have to be split into the different departure processes.

3.1.3 Aggregation

Suppose N arrival streams where every stream n is characterized by a rate λ_n and a SCV C_n^A. Every stream also has a service process, with mean time ES_n and SCV C_n^S. The aggregated processes can be approximated with the following equations [Whi83]:

$$\lambda = \sum_{n=1}^{N} \lambda_n \tag{3.4}$$

$$C^A = \frac{1}{\lambda} \sum_{i=1}^{N} \lambda_n C_n^A \tag{3.5}$$

$$ES = \frac{1}{\lambda} \sum_{i=0}^{N} \lambda_n ES_n \tag{3.6}$$

$$C^S = \left(\frac{1}{\lambda (ES)^2} \sum_{i=0}^{N} \lambda_n (ES_n)^2 \left(C_n^S + 1 \right) \right) - 1 \tag{3.7}$$

Where λ is the mean arrival rate, C^A the SCV of the arrival process, ES the mean service time, C^S the SCV of the service process.

After the departure process is approximated from the aggregated processes, it can be split into the N original processes. The mean departure rate ED_n is equal to the mean arrival rate of every stream n. For the SCV's C_n^D, the following approximation given by Whitt [Whi83] is used.

$$ED_n = \lambda_n \tag{3.8}$$

$$C_n^D = \frac{\lambda_n}{\lambda} C^D + 1 + \frac{\lambda_n}{\lambda} \tag{3.9}$$

where C^D is the SCV of the departure process of the aggregated stream.

3.1.4 Waiting time

With the arrival and service process of a queue the average waiting time (EW) can be calculated. For a $M/M/1$-queue the average waiting time is given by

$$EW_{M/M/1} = \frac{\rho}{1-\rho} ES \tag{3.10}$$

where ES is the mean service time of the queue.

For a $M/M/c$-queue the average waiting time is given by

$$EW_{M/M/c} = \frac{(c\rho)^c}{c!G} \frac{\rho}{(1-\rho)^2} \frac{ES}{c} \tag{3.11}$$

where G is a normalization constant, which can be obtained from the stationary distribution of the queue. Because of the computationally effort needed to obtain this G, equation 3.11 is not useful in the proposed iteration. [FWZ00] give an approximation for $EW_{M/M/c}$ which is easier to use:

$$EW_{M/M/c} = \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} ES \tag{3.12}$$

For queues with more general arrival and service time distributions the mean waiting time is given by

$$EW_{GI/G/c} = \frac{C^A + C^S}{2} EW_{M/M/c} \tag{3.13}$$

where C^S and C^A are the SCV of the service and arrival processes.

Another useful queue is a $M/M/c/K$-queue. This is an $M/M/c$-queue, where arrivals occur from a finite population. The Markov chain of this queue is given in figure 3.2

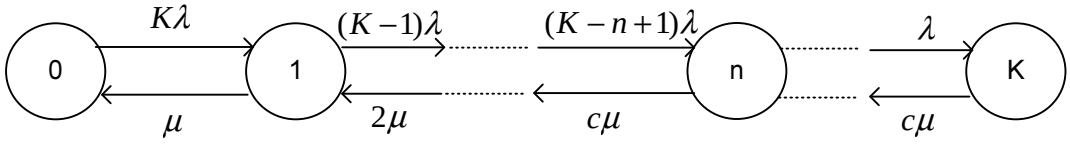

Figure 3.2: Markov chain of a $M/M/c/K$-queue

To obtain the stationary distribution π of this Markov chain, the following relation follows from the Markov chain:

$$\min(n, c)\mu\pi_n = (K - n + 1)\lambda\pi_{n-1} \tag{3.14}$$

This results in the following expression for π_n:

$$\begin{aligned} \pi_n &= \frac{(K-n+1)\lambda}{\min(n,c)\mu}\pi_{n-1} \\ &= \begin{cases} \binom{K}{n}\rho^n\pi_0 & if\ n \le c \\ \frac{K!}{c!c^{n-c}(K-n)!}\rho^n\pi_0 & if\ n > c \end{cases} \end{aligned} \tag{3.15}$$

where $\rho = \frac{\lambda}{\mu}$ and $\pi_0 = \left(\sum_{n=1}^{K} \pi_n\right)^{-1}$.

From this stationary distribution, the average number of jobs in the queue EL can be obtained.

$$\begin{aligned} EL &= \sum_{n=c+1}^{K} (n-c)\pi_n \\ &= \sum_{n=c+1}^{K} \frac{K!(n-c)\rho^n}{c!c^{n-c}(K-n)!}\pi_0 \end{aligned} \tag{3.16}$$

The average waiting time EW now follows from Little's law

$$EW = \frac{EL}{\lambda} \tag{3.17}$$

For the average waiting time in a queue with more general processes the same approximation as for a $GI/G/c$-queue is used

$$EW_{GI/G/c/K} = \frac{C^A + C^S}{2} EW_{M/M/c/K} \tag{3.18}$$

where C^S and C^A are the SCV of the service and arrival processes.

3.2 Operator queueing network

The network in figure 3.1 consists of three parts (queues): the machines, the operator queue and an ample server. For every part the departure process can be determined in the following way:

Machines This part consists of a synchronization station for every machine. In this synchronization station, arriving products to the machine have to be attached to a machine server. This combination product-machine is called an *operation*.
The two arrival processes at this queue are the arrival process of the products and the arrival process of free machine servers. The arrival processes of the different products can be aggregated to one arrival stream with equations (3.4) and (3.5). The arrival process of free machine servers is a single process. The average time between arrivals of this process is the total time the machine server is occupied. This is not only the total service time of the operation, it also includes the waiting time at the operator group.
In literature, nothing can be found about the departure process of a synchronization station. It can be stated that the arrival products are being served with the rate of the arrival of free machine servers. Therefore, we suggest to approximate the departure process of this synchronization station with the formulas given in 3.1.2, where the arrival process is the arrival process of the products, and the service process is the arrival process of the free machine servers.
In the same way, the mean waiting time of products can be obtained using equation (3.13). The SCV of this process is the aggregated SCV of the service time with the operator and the service time at the ample server. This holds for every operation on the machine and these service processes can be aggregated to one service process with equations (3.6) and (3.7).

The mean departure rate is equal to the mean arrival rate of products, and equation (3.3) can be used to approximate the SCV of the departure process of this queue.

Operator queue This is a $G/G/p$-queue where every operation is a different class. These different classes arrive from a finite population (the total number of machines is finite) and are aggregated per machine. These can be aggregated to one input stream using equations (3.4) and (3.5).
The service time of a product class in this station is the time an operator is needed. This is know for every operation. The aggregated service process can be obtained using equations (3.6) and (3.7).

The mean departure rate equals the mean arrival rate of operations. Using the aggregated arrival and service process, equation (3.3) gives an approximation of the SCV of the departure process. This SCV can be split into an SCV for every operation with equation (3.9).

Ample server This is a queue with no waiting time. Here also every operation is a different class, and again the arrive from a finite population. A queue with no waiting time can be obtained when taking as many servers as the size of the total population, which is the total number of machine servers for all the machines (K). Then this queue becomes a $GI/G/K/K$-queue.

In the same way as for the operator queue, the departure process can be obtained. Only for the service time of an operation, the time where no operator is necessary has to be used.

When products (operations) leave the ample server, they leave the system or they move to the next machine according their stochastic routing through the network. Therefore, the arrival process of products at the machine queues depends on the departure process of the ample server. The mean arrival rate of every product to a machine (operation) is now the external arrival plus the fraction of products that move to the machine from other machines. In a formula this will be

$$\lambda_{rm} = \lambda_{rm}^E + \sum_{i=1}^{M} T_{im}^r \lambda_{ri} \tag{3.19}$$

where λ_{rm} is the mean arrival rate of product r to machine m, λ_{rm}^E the external arrival rate of product r to machine m and T^r the routing matrix of product r. Writing this equation down for every product r and machine m gives equations (3.20) for every product r.

$$\vec{\lambda}_r = \vec{\lambda}_r^E + T^r \vec{\lambda}_r \tag{3.20}$$

The solution of this matrix equation gives all the arrival rates of products at the machines.

Based on [BS93], an expression for the SCV of the arrival process is given by

$$C_{rm}^A = \frac{\lambda_{rm}^E}{\lambda_{rm}} + \frac{1}{\lambda_{rm}} \sum_{j=1;j\neq m}^{M} \lambda_{rj} T_{jm}^r \left(T_{jm}^r C_{rj}^D + (1 - T_{jm}^r) \right) \tag{3.21}$$

where C_{rm}^A is the SCV of the arrival process of product r to machine m and C_{rm}^D the SCV of the departure process of operation rm of the ample server.

When all the departure processes are known, the total waiting time of every operation EW_{rm} can be computed. This is the sum of the average waiting time at the machine queue (EW_m) and the average waiting time at the operator queue (EW_{opp}). The machine queue is a $GI/G/c_m$-queue, thus (3.13) can be used to find the average waiting time.
The operator queue is a $GI/G/p/K$-queue, with $K = \sum_{i=1}^{M} c_i$. The average waiting time can be found with (3.18). For large values of p and K, this expression for the average waiting of a $GI/G/p/K$-queue can be complex to calculate. Then an approximation has to be used. Because no approximation is know by the author, and it will only be a problem for very large values of c and K, (3.18) will be used in the new algorithm.

This leads to the following formula for the total average waiting time of operation rm

$$\begin{aligned} EW_{rm} &= EWm + EWopp \\ &= EW_{GI/G/c_m} + EW_{GI/G/p/K} \\ &= \frac{C_m^A + C_m^S}{2} EW_{M/M/c_m} + \frac{Cp^A + Cp^S}{2} EW_{M/M/p/K} \\ &= \frac{C_m^A + C_m^S}{2} \frac{(\lambda_m ES_m)^{\sqrt{2(c_m+1)}-1}}{c_m(1-\lambda_m ES)} ES_m + \\ &\quad \frac{Cp^A + Cp^S}{2\lambda} \sum_{n=p+1}^{K} \frac{K!(n-p)\left(\lambda_p ES_p\right)^n}{p!p^{n-p}(K-n)!} \pi_0 \\ &= \frac{C_m^A + C_m^S}{2} \frac{(\lambda_m ES_m)^{\sqrt{2(c_m+1)}-1}}{c_m(1-\lambda_m ES)} ES_m + \\ &\quad \frac{1}{G} \frac{Cp^A + Cp^S}{2\lambda} \sum_{n=p+1}^{K} \frac{K!(n-p)\left(\lambda_p ES_p\right)^n}{p!p^{n-p}(K-n)!} \end{aligned} \tag{3.22}$$

where

$$G = \sum_{n=0}^{p} \binom{K}{n} \left(\lambda_p ES_p\right)^n + \sum_{n=p+1}^{K} \frac{K!\left(\lambda_p ES_p\right)^n}{p!p^{n-p}(K-n)!}, \tag{3.23}$$

and where C_m^A and Cp^A are the SCV's of the arrival processes at machine m and the arrival process at the operator queue, C_m^S and Cp^S are the SCV's of the service processes at machine m and the service process at the operator group. K is the total number of machine servers in the network, which is the maximum size of the population from which operations arrive at the operator queue. With these waiting times, the average queue lengths, and the sojourn time of a product in the network can be obtained.

3.2.1 Iteration

Now all parts of the iteration are specified, a formal algorithm to find departure processes and waiting times in a OQN can be made.
Because not every operation will visit the operator group and the ample server, some equations do not have to be computed for every operation. To formalize this, the variables a_{rm} and b_{rm} are introduced. If operation rm visits the operator group then $a_{rm} = 1$ and $a_{rm} = 0$ if does not visit the operator group. If operation rm visits the ample server then $b_{rm} = 1$ and $b_{rm} = 0$ if it does not. These variables are included in the appropriate equations.

All this leads to algorithm 1. An explanation of the used variables can be found in appendix A

Algorithm 1

Algorithm to find departure processes and waiting times in a OQN with a single operator group and exponential external arrivals.

1. (Initialize) *Set* $EW_{opp} = 0$. *For* $m = 1 \dots M$, $C_m^S = 1$.
   For $r = 1 \dots R$ *and* $m = 1 \dots M$ *set* $C_{rm}^A = 1$ *and let*

$$a_{rm} = \begin{cases} 0 & if\ ES_{rm}^{opp} = 0 \\ 1 & if\ ES_{rm}^{opp} > 0 \end{cases}$$

$$b_{rm} = \begin{cases} 0 & if\ ES_{rm}^{amp} = 0 \\ 1 & if\ ES_{rm}^{amp} > 0 \end{cases}$$

2. *For* $r = 1 \dots R$ *compute the arrival rate of products at the machines* λ_{rm} *from*

$$\vec{\lambda}_r = \vec{\lambda}_r^E + T^r \vec{\lambda}_r$$

3. *Set the aggregated arrival rates and service times for the parts of the network*
   For $m = 1 \dots M$

$$\lambda_m = \sum_{r=1}^{R} \lambda_{rm}$$

   and

$$\lambda_p = \sum_{r=1}^{R} \sum_{m=1}^{M} a_{rm} \lambda_{rm}$$

$$ES_p = \sum_{r=1}^{R} \sum_{m=1}^{M} a_{rm} \frac{\lambda_{rm}}{\lambda_p} ES_{rm}^{Opp}$$

$$\lambda_{amp} = \sum_{r=1}^{R} \sum_{m=1}^{M} b_{rm} \lambda_{rm}$$

$$ES_{amp} = \sum_{r=1}^{R} \sum_{m=1}^{M} b_{rm} \frac{\lambda_{rm}}{\lambda_p} ES_{rm}^{amp}$$

4. *repeat*

   (a) *Set the service process of the products at the machines.*
   For $r = 1 \dots R$ *and* $m = 1 \dots M$

$$ES_{rm} = a_{rm} EW_{opp} + ES_{rm}^{opp} + ES_{rm}^{amp}$$

$$C_{rm}^S = \frac{a_{rm} ES_{rm}^{opp}\, Cs_{rm}^{opp} + b_{rm} ES_{rm}^{amp}\, Cs_{rm}^{amp}}{a_{rm} ES_{rm}^{opp} + b_{rm} ES_{rm}^{amp}}$$

(b) *Set the aggregated arrival and service processes for the machines.*
For $m = 1 \dots M$

$$C_m^A = \sum_{r=1}^{R} \frac{\lambda_{rm}}{\lambda_m} C_{rm}^A$$

$$ES_m = \sum_{r=1}^{R} \frac{\lambda_{rm}}{\lambda_m} ES_{rm}$$

$$C_m^S = \sum_{r=1}^{R} \frac{\lambda_{rm}}{\lambda_m} C_{rm}^S$$

(c) *Get the SCV of the departure process of the machines.*
For $m = 1 \dots M$

$$C_m^D = 1 + \frac{(\lambda_m ES_m)^2}{\sqrt{c_m}} \left(C_m^S - 1\right) + \left(1 - (\lambda_m ES_m)^2\right) \left(C_m^A - 1\right)$$

(d) *Split the SCV of the departure process of the machines.*
For $r = 1 \dots R$ *and* $m = 1 \dots M$

$$C_{rm}^D = \frac{\lambda_{rm}}{\lambda_m} C_m^D + 1 + \frac{\lambda_{rm}}{\lambda_m}$$

(e) *Get the departure processes of the operator group.*
For $r = 1 \dots R$ *and* $m = 1 \dots M$, *if* $a_{rm} = 1$

i. Set the SCV's of the aggregated arrival and service processes.

$$Cp^A = \sum_{r=1}^{R} \sum_{m=1}^{M} a_{rm} \frac{\lambda_{rm}}{\lambda_p} C_{rm}^D$$

$$Cp^S = \left(\frac{1}{\lambda_p (ES_p)^2} \sum_{r=1}^{R} \sum_{m=1}^{M} a_{rm} \lambda_{rm} (ES_{rm}^{Opp})^2 \left(Cs_{rm}^{Opp} + 1\right) \right) - 1$$

ii. Get the SCV of the aggregated departure process.

$$Cp^D = 1 + \frac{(\lambda_p ES_p)^2}{\sqrt{p}} \left(Cp^S - 1\right) + \left(1 - (\lambda_p ES_p)^2\right) \left(Cp^A - 1\right)$$

iii. Split the SCV of the aggregated departure process.
For $r = 1 \dots R$ *and* $m = 1 \dots M$

$$C_{rm}^D = \frac{\lambda_{rm}}{\lambda_p} Cp^D + 1 + \frac{\lambda_{rm}}{\lambda_p}$$

(f) *Get the departure processes of the ample server.*
For $r = 1 \dots R$ *and* $m = 1 \dots M$, *if* $b_{rm} = 1$

i. Set the SCV's of the aggregated arrival and service processes.

$$Ca^A = \sum_{r=1}^{R} \sum_{m=1}^{M} b_{rm} \frac{\lambda_{rm}}{\lambda_a mp} C_{rm}^D$$

$$Ca^S = \left(\frac{1}{\lambda_{amp} (ES_{amp})^2} \sum_{r=1}^{R} \sum_{m=1}^{M} b_{rm} \lambda_{rm} (ES_{rm}^{amp})^2 \left(Cs_{rm}^{amp} + 1\right) \right) - 1$$

ii. Get the SCV of the aggregated departure process.

$$Ca^D = \left(1 - (\lambda_{amp} ES_{amp})^2\right) Ca^A + (\lambda_{amp} ES_{amp})^2 \, Ca^S$$

iii. Split the SCV of the aggregated departure process.
For $r = 1 \dots R$ and $m = 1 \dots M$

$$C^D_{rm} = \frac{\lambda_{rm}}{\lambda_{amp}} Ca^D + 1 + \frac{\lambda_{rm}}{\lambda_{amp}}$$

(g) Set the waiting time at the operator queue.

$$G = \sum_{n=0}^{p} \binom{K}{n} (\lambda_p ES_p)^n + \sum_{n=p+1}^{K} \frac{K! \, (\lambda_p ES_p)^n}{p! p^{n-p} (K-n)!}$$

$$EW_{opp} = \frac{1}{G} \frac{Cp^A + Cp^S}{2\lambda} \sum_{n=p+1}^{K} \frac{K!(n-p) \, (\lambda_p ES_p)^n}{p! p^{n-p} (K-n)!}$$

(h) Reset the SCV's of the arrival process of products at the machines.
For $r = 1 \dots R$ and $m = 1 \dots M$

$$C^A_{rm} = \frac{\lambda^E_{rm}}{\lambda_{rm}} + \frac{1}{\lambda_{rm}} \sum_{j=1; j \neq m}^{M} \lambda_{rj} T^r_{jm} \left(T^r_{jm} C^D_{rj} + (1 - T^r_{jm})\right)$$

until convergence of all C^D_{rm}

5. *Obtain the waiting time of the operations.*
For $r = 1 \dots R$ and $m = 1 \dots M$

$$EW_{rm} = EW_{opp} + \frac{C^A_m + C^S_m}{2} \frac{(\lambda_m ES_m)^{\sqrt{2(c_m+1)}-1}}{c_m (1 - \lambda_m ES_m)} ES_m$$

3.3 Operators in a MC GKCS

The algorithm developed in the previous sections can easily be extended to other types of networks. Four types of networks are considered: an OQN, a *closed queueing network* (CQN), a *semi open queueing network* (SOQN) and a kanban system.
The difference between an OQN and a CQN or a SOQN is that in the last two, the capacity of the network is restricted by a fixed number of pallets circling through the network. Products arriving to the network are attached to a pallet before they enter it. The product stays attached to this pallet until it leaves the network. If no pallet is available, the product has to wait. When the product leaves the network, the pallet becomes available for a next product.
The difference between a CQN and a SOQN is the arrival process at the pallet queue. In a CQN there is an infinite amount of products waiting, thus a pallet directly can pick a new product when its empty. In a SOQN products arrive at this queue according to some stochastic process, and thus an empty pallet might have to wait before it can pick a new

product. Note that when the number of pallets is set to ∞ we have the previous network for an OQN.

Algorithm 1 can easily be extended to analyze a CQN or a SOQN. An extra synchronization station is added for every product. The external arrival process of products is replaced by the departure process of the new synchronization station. The service process of the new synchronization station is the total service process of the corresponding product in the network, which is the sum of all service times of every operation plus the corresponding waiting times.
The arrival process at the new synchronization stations equals the external arrival process. For a CQN this is set to infinity.
All these extra synchronization stations can be analyzed with the same equations as used in the previous section.

In a kanban system, one cell can be described with the network given in figure 3.1. For every product, two synchronization stations are added for every cell. One to combine external demand for the network and an item from base stock, and one to combine a production request and a kanban. The departure process of the first new synchronization station is the external arrival at the cell. The departure process of products from the cell is the service process of the first synchronization station. The arrival process of the first synchronization station is the external arrival process of products to the network, if the cell is the last visit of the product. If the cell is not the last visit, the arrival process of the first synchronization station is the departure process of the second synchronization station of the next cell the product visits.
The arrival process to the second synchronization station is the same as the service process of the first synchronization station. The service process of the second synchronization stationequals the arrival process of the first synchronization station.

The *market service levels* are interesting performance measures in a kanban system. These can still be obtained from the analysis of the last synchronization station of a visit, using the equations in appendix C.

3.4 Remarks

Some remarks can be made about algorithm 1:

- Algorithm 1 should also work if there are no operators. The operator queue then becomes a $GI/G/\infty/K = GI/G/K/K$ queue, there is no waiting time at this queue.
- Algorithm 1 can easily be extended for multiple operator groups.
- The need for an operator is assumed to be only at the beginning of the operation time. This can be changed into other combinations very easy.
- In general, the external arrival to the network is assumed to be exponential distributed. Because algorithm 1 also uses the SCV of the external arrival process, other situations can be handled.

Chapter 4

Test results

To see if the approximation of the new algorithm works well, two simple examples are analyzed analytical and with the new algorithm and the results are compared.

Example 1

Suppose the queuing network given in figure 4.1. This network consists of one machine with two servers, one product and one operator. Products arrive at the queue with rate λ. When a product arrives at the machine, it occupies a server, if one is free. In that case, they wait together for the operator to become available. When this operator is free, the product is served. For the whole serving time ES, the operator care is needed. After this, the product leaves the system, and the operator and the machine become available again. In figure 4.2 this network is given with the machine changed into a synchronization station.

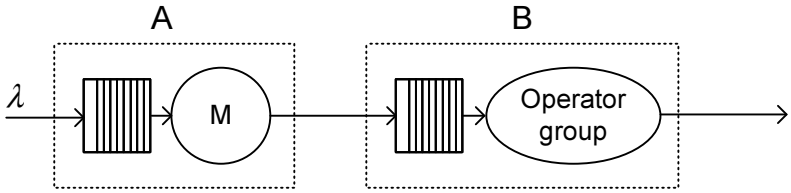

Figure 4.1: Queuing network 1

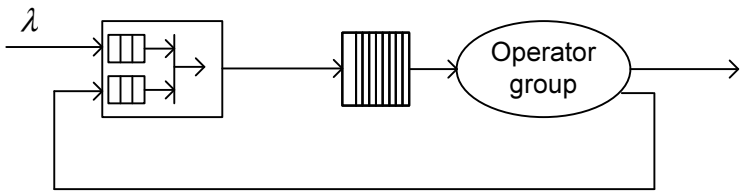

Figure 4.2: Queuing network 1 with machine synchronization station

For the Markov chain of this system, as depicted in figure 4.3, the states are given by the number of products in the first queue (A) and the number of products in the second queue

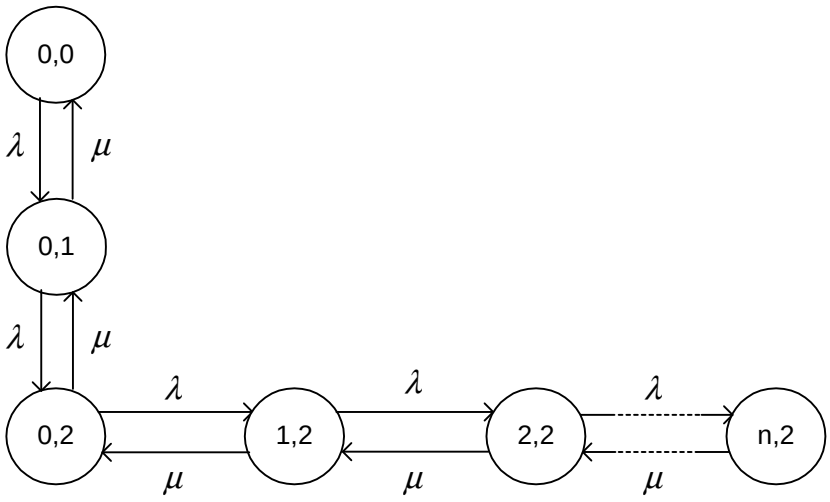

Figure 4.3: Markov chain for network 1

(B). When looking at this Markov chain, the network is a simple $M/M/1$-queue. For a $M/M/1$-queue, the average waiting time equals the following formula:

$$EW = \frac{\rho}{1-\rho} ES \tag{4.1}$$

Let $\lambda = 2$ and $\mu = \frac{1}{3}$. Then, the average waiting time of the product in the network equals $\frac{2}{3}$.
Algorithm 1 results in a average waiting time of $0,4571$ at the operator queue, and an average waiting time of $0,2776$ in the machine queue. These results are obtained after two iteration steps. This gives a total average waiting time of $0,7347$.

Because there is no routing back of products, the arrival process of products at the machine queue does not change after the first step of the iteration. Therefore, the results do not change in the second step of the iteration, and thus the total number of steps before convergence is only two.
The total average waiting time obtained with algorithm 1 has an error of 9.3% compared with the analytical solution.

Example 2

Suppose the same system as in example 1, but now the operator is only needed to setup the product on the machine. Thus when the operator is finished, the product is being served for the rest of the time and the operator picks a new product/server from its queue (figure 4.4). The rate of which the operator sets up the products is called μ_{Opp} and the rate at which products are served at the ample server is called μ_M.

This problem is more difficult to analyze. There are three places where products can stay: the queue for the machine (A), the (queue for the) operator group (B) and the machine during service (C). Call the number of products in A, B and C respectively a, b, c.
b and c can only have the values $0, 1, 2$. Also the sum of this can not exceed two, because to

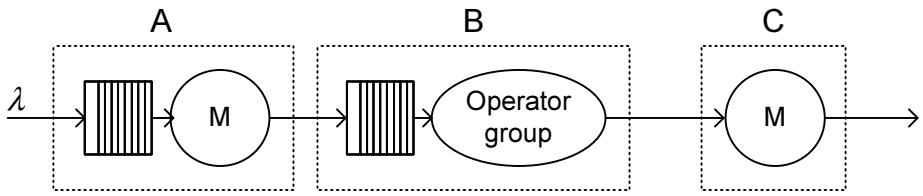

Figure 4.4: Queuing network 2

all products in these queues, a machine server is attached, and there are only two machine servers. The states of the Markov chain consists of the number of products in every part (a, b and c). This Markov chain is given in figure 4.5.

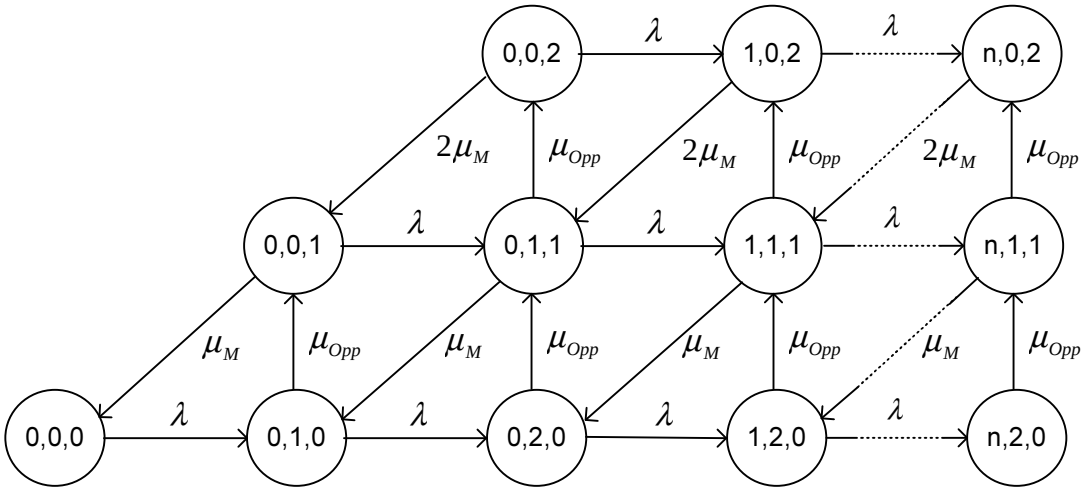

Figure 4.5: Markov chain for network 2

The stationary distribution π of a Markov process is characterized as the unique solution of $\pi Q = 0$ and $\pi e = 1$, where Q is the generator matrix consisting of the transition rates and e denotes the column vector with appropriate size of which all elements are equal to one. The analytical solution for the stationary distribution of this Markov chain is given by Buitenhek [Bui98]. He obtained a Markov chain with the same structure for a different problem and solved it with the matrix-geometric method described by Neuts [Neu81].

The state space of the Markov process consists of levels, $l = 0, 1, 2, \ldots$, where each level l is defined as the set of states with $a = l$. For $l > 0$ the states are ordered as $(l, 0, 2), (l, 1, 1), (l, 2, 0)$. For $l = 0$ the states are ordered as $(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 0, 2), (0, 1, 1), (0, 2, 0)$. Now the different parts of Q can be written down.
For $l > 0$, let A_0 be the matrix with transition rates from level l to level $l + 1$, A_1 the

transition rates inside level l and A_2 the transition rates from level l to level $l-1$. Then

$$
\begin{aligned}
A_0 &= \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix} \\
A_1 &= \begin{pmatrix} -(\lambda+\mu_{Opp}) & \mu_{Opp} & 0 \\ 0 & -(\lambda+\mu_{Opp}+2\mu_M) & \mu_{Opp} \\ 0 & 0 & -(\lambda+\mu_M) \end{pmatrix} \\
A_2 &= \begin{pmatrix} 0 & 0 & 0 \\ 2\mu_M & 0 & 0 \\ 0 & \mu_M & 0 \end{pmatrix}
\end{aligned}
$$

Let B_0 be the matrix with transition rates from level 0 to level 1, B_1 the transition rates inside level 0, and B_2 the transition rates from level 1 to level 0. Then

$$
\begin{aligned}
B_0 &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix} \\
B_1 &= \begin{pmatrix} -\lambda & 0 & \lambda & 0 & 0 & 0 \\ \mu_M & -(\lambda+\mu_M) & 0 & 0 & \lambda & 0 \\ 0 & \mu_{Opp} & -(\lambda+\mu_{Opp}) & 0 & 0 & \lambda \\ 0 & 2\mu_M & 0 & -(\lambda+2\mu_M) & 0 & 0 \\ 0 & 0 & \mu_M & \mu_{Opp} & -(\lambda+\mu_M+\mu_{Opp}) & 0 \\ 0 & 0 & 0 & 0 & \mu_{Opp} & -(\lambda+\mu_{Opp}) \end{pmatrix} \\
B_2 &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu_M & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu_M & 0 \end{pmatrix}
\end{aligned}
$$

De generator matrix can now be written as

$$
Q = \begin{pmatrix} B_1 & B_0 & 0 & 0 & 0 & \\ B_2 & A_1 & A_0 & 0 & 0 & \dots \\ 0 & A_2 & A_1 & A_0 & 0 & \\ 0 & 0 & A_2 & A_1 & A_0 & \\ & & \vdots & & & \ddots \end{pmatrix} \tag{4.2}
$$

For $l > 0$, let $\vec{\pi_l}$ the stationary distribution vector of the states in level l, and let $\vec{\pi_0} = (\pi_{(0,0,2)}, \pi_{(0,1,1)}, \pi_{(0,2,0)})$. It can be shown [Neu81] that a non-negative matrix $R \in \mathbb{R}^{3\times 3}$ exists for which

$$
\vec{\pi_l} = \vec{\pi_0} R^l \tag{4.3}
$$

and R is the solution of the quadratic matrix equation

$$
R = A_0 + R\,A_1 + R^2 A_2 \tag{4.4}
$$

The solution of this equation can be found with repeated substitution, starting with $R = \mathbf{0}$.

The average number of jobs in the queue (EL) is given by

$$\begin{aligned} EL &= \sum_{l=1}^{\infty} l\, \vec{\pi_0} R^l e \\ &= \vec{\pi_0}\, R(I-R)^{-2} e \end{aligned} \tag{4.5}$$

The values for the unknown stationary probabilities $\vec{\pi_0}$ can be found by solving the balance equations for the states in level 0 and using the normalization condition

$$\sum_{n=0}^{5} \pi_{0,n} + \sum_{l=1}^{\infty} \sum_{n=0}^{2} \pi_{l,n} = 1 \tag{4.6}$$

The average waiting time (EW) is given by

$$EW = \frac{EL}{\lambda} \tag{4.7}$$

Now take $\lambda = 2$, $\mu_{opp} = 9$ and $\mu_M = 4,5$. Filling in these values, and evaluating equation (4.4) results in

$$R = \begin{pmatrix} 0,5067 & 0,05851 & 0,006501 \\ 0,03020 & 0,5164 & 0,05738 \\ 0,006694 & 0,05851 & 0,5065 \end{pmatrix} \tag{4.8}$$

Solving the balance equations of the states in level 0 results in

$$\vec{\pi_0} = \begin{pmatrix} 0,2202 & 0,1742 & 0,2202 & 0,1426 & 0,08233 & 0,02744 \end{pmatrix} \tag{4.9}$$

Filling in the formula for EW results in an average waiting time of $0,4252$.
Algorithm 1 results in a average waiting time of $0,0637$ at the operator queue, and an average waiting time of $0,2778$ in the machine queue. This gives a total average waiting time of $0,3414$. This has an error of 25.0% compared with the analytical solution.

Chapter 5

Conclusion

5.1 Conclusions

An iterative algorithm is developed to analyze different types of networks with operators: OQN, SOQN, CQN and kanban systems. This new algorithm is tested in two small networks. For these networks, the markov chains could be solved analytically. With this analytical solution, the new algorithm could be validated.
The relative error of the new method is quite large, but with small numbers, the absolute error is more important, because even a small error is a large fraction of the analytical solution. This absolute error is not so big, and the results show that the new method might be useful for larger networks.

5.2 Further research

The method given in this report has to be validated for larger networks, especially kanban systems. For this, simulation models have to be made and the results of the new method have to be compared with the simulation results.
Many authors have given approximations for the aggregation of processes or obtaining the departure process. The new method can possibly be improved by taking better approximations for these processes. But in general: the better the approximation, the more complex it is to obtain. Thus better approximations will increase the runtime of the algorithm.
A problem for larger networks can be the currently used formula for the average waiting time in the operator queue. For a large number of operators and machines, it can be difficult to compute the normalization constant in this formula. It is best to find an approximation for the average waiting time in a $GI/G/c/K$-queue. Also an approximation for the binomial coefficient in the summation will solve the problem, because this part is the most complex to compute for large numbers.

Another important issue to investigate is the time complexity of the new method. If the new model gives good approximations of the performance measures, and it is faster then currently used models (for networks without operators), it can be used in optimization of larger networks. This optimization, or *parameter setting* is an important aspect of the modeling of manufacturing systems. The best combination of several parameters has to be found. But when is a combination the best, and what parameters can be controlled in an existing manufacturing system?

The best combination is the cheapest. For all controllable parameters holds: the higher a parameter, the higher the cost. The base stock level uses capacity, an extra operator have to be payed. Thus these parameters have to be minimized, but still some performance level has to be reached, for instance a fill rate of 95%.
There are several parameters that can be controlled and for wich it is not obvious what the value must be. An example of a parameter with an obvious value is the variability of the service time. This have to be reduced as much as possible. The most important controllable parameters are the number of kanbans N, the base stock level S and the number of operators P in every operator group.
At the moment this optimization is only possible with running an algorithm for a lot of situations and pick the one with best results. For real manufacturing systems one run of MC GKCS needs a lot of time. Therefore optimization will take years, which is not a suitable solution.

Not much can be said about the change of the performance measures when one of the three parameters changes. Earlier work [Wor01] showed that this is quite unpredictable. He computed the performance measures for lot of cases, and found no relations with the parameters, not even a direction. A question arising from this: Is this a property of the model, or a property of the manufacturing system? This can only be checked with simulation results. With this simulation, it can be found if there will be a global trend or not. In that case, this global trend can be used to eliminate parts of the state space of parameter values.
Another way to decrease the possible solutions is to find upper and lower bounds for the parameters using other models (i.e. find optimal solutions for N or S using optimizing models for classic kanban system and base stock models). The relations between N and S can then be used to obtain better bounds.
For the operators an upper bound is the total number of machines. With this number of operators an arriving machine never has to wait. Having more operators does not decrease the sojourn time in the system. A lower bound can be found from the stability condition and the total arrival rate at the operator queue.

An interesting question is if the new model is faster in optimization methods. In an optimization method, performance measures of a network have to be computed for different situations, to find the best parameter set. The AMVA procedure used in the MC GKCS-algorithm of Wormgoor [Wor01] has to start with zero in every run. The new model can start with the results obtained with a previous parameter set. Therefore every run after the first will be much faster, and reduce the total time the optimization method needs. Even if the approximation is not very accurate, maybe some parts of the state space of the parameters can be eliminated, which also reduces the time needed by the optimization method.

Bibliography

[AVW98] Ivo Adan, Jeremy Visschers, and Jaap Wessels. Sum of product forms solutions to msccc queues with job type dependent processing times. Memorandum COSOR 98-19, Eindhoven University of Technology, 1998.

[BD96] B. Baynat and Y. Dallery. A product-form approximation method for general closed queueing networks with several classes of customers. *Performance Evaluation*, 24:165188, 1996.

[BDMF01] B. Baynat, Y. Dallery, M. Di Mascolo, and Y. Frein. A multi-class approximation technique for the analysis of kanban-like control systems. *International Journal of Production Research*, 2, January 2001. Special Issue on Modeling, Specification and Analysis of Manufacturing Systems.

[BS93] J.A. Buzacott and J.G. Shanthikumar. *Stochastic Models of Manfacturing Systems*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Bui95] R. Buitenhek. Manufacturing systems: Performance evaluation and due date assignment. Report for design of mathematical control and managment models, University of Twente, March 1995.

[Bui98] R. Buitenhek. *Performance evaluation of dual resource manufacturing systems*. PhD thesis, University of Twente, Enschede, 1998.

[BvHZ99] R. Buitenhek, G.J. van Houtum, and W.H.M. Zijm. An open queueing model for flexible manufacturing systems with multiple part types and general purpose pallets. In J. Ashayeri, W.G. Sullivan, and M.M. Ahmad, editors, *Flexible Automation and Intelligent Manufacturing*, pages 713–734. Begell House, New York, 1999.

[BvHZ00] R. Buitenhek, G.J. van Houtum, and W.H.M. Zijm. Amva based solution procedures for open queueing networks with population constraints. *Annals of Operations Research*, 93:15–40, 2000.

[CA03] Srinivas R. Chakravarthy and Atul Agarwal. Analysis of a machine repair problem with an unreliable server and phase type repairs and services. In preparation for Naval Research Logistics, 2003.

[Chi01] W.K. Ching. Machine repairing models for production systems. *International Journal of Production Economics*, 70(3):257–266, April 2001.

[CHSZ02] T. Coenen, S.S. Heragu, M. Srinivasan, and H. Zijm. Analysis of manufacturing systems via semi-open queuing networks for multiple classes. Technical report, Decision Sciences and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy, NY, 2002.

[Dal90] Y. Dallery. Approximate analysis of general open queueing networks with restricted capacity. *Performance Evaluation*, 11:209–222, 1990.

[DHMO89] J.L. Deleersnyder, T.J. Hodgson, H. Muller, and P.J. O'Grady. Kanban controlled pull systems: An analytic approach. *Management Science*, 35:1079–1091, 1989.

[DS96] P. Desruelle and H. J. Steudel. Queuing network model of a single-operator manufacturing workcell with machine/operator interference. *Management Science*, 42(4):576–590, 1996.

[Ebe98] M. Eben-Chaime. Spreadsheet calculations for the machine interference queueing model. *Computers & Industrial Engineering*, 34(2):385–390, April 1998.

[FWZ00] N.D. van Foreest, O.S. Wormgoor, and W.H.M. Zijm. Manufacturing and logistic systems analysis, planning and control. Syllabus University of Twente, 2000.

[Gol01] H Gold. A simple queueing model for the estimation of man machine interference in semiconductor wafer fabrication. In *Operations Research proceedings 2001*, Duisburg, September 2001.

[GY94] P. Glasserman and D.D. Yao. *Monotone Structure in Discrete-Event Systems*. Series in Probability and Mathematical Statistics. Wiley Inter-Science, 1994.

[KK89] U.S. Karmarkar and S. Kekre. Batching policy in kanban systems. *Journal of Manufacturing Systems*, 8:317–328, 1989.

[KK99] T. Kurasugi and I. Kino. Approximation methods for two-layer queueing models. *Performance Evaluation*, 36/37:55–69, August 1999.

[Klo03] M.P. van 't Klooster. Performance evaluation of Multi-Class Generalized Kanban Controlled Systems, implementation in a new Decision Support System. Report on practical training, University of Twente, 2003.

[LBR89] P. Legato, A. Bobbio, and L. Roberti. The effect of failures and repairs on multiple cell production lines. In *Proceedings of 20th International Symposium on Automotive Technology and Automation*, Florence, Italy, 1989.

[Mar79] R. Marie. An approximate method for general queueing networks. *IEEE Transactions on Software Engineering*, 5:530–538, 1979.

[MFBD93] M. Di Mascolo, Y. Frein, B. Baynat, and Y. Dallery. Queueing network modelling and analysis of generalized kanban systems. In *Proceedings of the Second European Control Conference (ECC93)*, pages 170–175, Groningen, the Netherlands, 1993.

[MFD96] M. Di Mascolo, Y. Frein, and Y. Dallery. An analytical method for performance evaluation of kanban controlled production systems. *Operations Research*, 44:50–64, 1996.

[MK97] M. Mittler and C. Kern. Discrete-time approximation of the machine interference problem with generally distributed failure, repair, and walking times. *European Journal on Control*, 3(4):254–267, 1997.

[MM90] D. Mitra and I. Mitrani. Analysis of a kanban discipline for cell coordination in production lines. i. *Management Science*, 36:1548–1566, 1990.

[MM91] D. Mitra and I. Mitrani. Analysis of a kanban discipline for cell coordination in production lines. ii: Stochastic demands. *Operations Research*, 35:807–823, 1991.

[Neu81] M. Neuts. Matrix-geometric solutions in stochastic models - an algorithmic approach. The Johns Hopkins University Press, 1981.

[PC99] C. G. Panayiotou and C. G. Cassandras. Optimization of kanban-based manufacturing systems. Working paper Department of Manufacturing Engineering, Boston University, 1999.

[SA85] K.E. Stecke and J.E. Anderson. Review of operator/machine interference models. *International Journal of Production Research*, 23(1):129–151, 1985.

[SB93] J. Sztrik and B.D. Bunday. Machine interference problem with a random environment. *European Journal of Operational Research*, 65(2):259–269, 1993.

[SDD86] R. Suri, G.W. Diehl, and R. Dean. Quick and easy manufacturing systems analysis using manuplan. In *Proceedings Spring IIE Conference*, pages 195–205, 1986.

[SdT91] R. Suri and S. de Treville. Full speed ahead: A look at rapid modeling technology in operations management. *OR/MS Today*, 18:295–304, 1991.

[SH02] Mukund Srinivasan and Sunderesh S. Heragu. Analysis of manufacturing systems via semi-open queuing networks. Technical Report 38-02-494, Decision Sciences and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy, NY, 2002.

[So89] K.C. So. Allocating buffer storage in a flexible manufacturing system. *International Journal Of Flexible Manufacturing Systems*, 1(3):223–233, 1989.

[SSK93] R. Suri, J.L. Sanders, and M. Kamath. Performance evaluation of production networks. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors, *Logistics of Production and Inventory*, chapter 5. Elsevier Science Publishers B.V., 1993.

[Szt01] J. Sztrik. Finite source queueing systems and their applications. http://it.math.klte.hu/user/jsztrik/research/fsqreview.pdf, 2001.

[Whi83] W. Whitt. The queueing network analyzer. *The Bell System Technical Journal*, 62(9), November 1983.

[Wor01] O.S. Wormgoor. Performance evaluation of generalized kanban systems. Master's thesis, University of Twente, Enschede, 2001.

Appendix A

List of symbols

M	Number of machines in the network
R	Number of products in the network
c_m	Number of machine servers of machine m
T^r	Route matrix of product r
$\vec{\lambda}_r$	Vector with arrivals per machine for a product
$\vec{\lambda}_r^E$	Vector with external arrivals per machine for a product
λ_{rm}	Arrival rate of product r at machine m
λ_m	Total arrival rate of products at machine m
λ_p	Total arrival rate of operations at the operator queue
λ_{amp}	Total arrival rate of operations at the ample server
ES_{rm}	Total service time of operation rm
ES_{rm}^{Opp}	Service time of operation rm during which an operator is needed
ES_{rm}^{Amp}	Service time of operation rm during which no operator is needed
ES_m	Total service time of machine m
ES_p	Total service time for the operator group
ES_{amp}	Total service time for the ample server
C_{rm}^S	Total SCV of the service process of operation rm
C_m^S	SCV of the service process of free machine servers at synchronization station m
C_{rm}^A	SCV of the arrival process of operation rm
C_m^A	SCV of the arrival process of machine m
C_m^D	SCV of the departure process of machine m

C_{rm}^D	SCV of the departure process of operation rm
Cp^A	SCV of the arrival process at the operator queue
Cp^S	SCV of the service process at the operator queue
Cp^D	SCV of the departure process of the operator queue
Ca^A	SCV of the arrival process at the ample server
Ca^S	SCV of the service process at the ample server
Ca^D	SCV of the departure process of the ample server
EW_{opp}	Average waiting time in the operator queue
EW_{rm}	Average waiting time in the network of operation rm

Appendix B

MC GKCS algorithm

For the analysis of a MC GKCS the first two iterative algorithms in this appendix are used, the fist to analyze a stage and the second for the whole network. The last algorithm is the AMVA-algorithm used to analyze the complement network of a synchronization station. All these algorithms are found in [Wor01].

Algorithm B.1 *Analysis of a multi-class GKCS*

1. *Set* $TH_i^{C_b^{(r)}}$ *to* $\lambda_E^{(r)}$ *for* $i = 1, ..., Z-1$, $r = 1, ..., R$
2. *Repeat steps* (a) *and* (b) *until convergence of the service rates of all synchronization stations*
   - *(a)* Upstream walk: *For stages* $i = Z, ..., 2$ *do*
     - *i. Use Algorithm B.2 to obtain the service rates of the synchronization stations* $J_{k,i-1}^{(r)}$ *and* $J_{b,i}^{(r)}$, $r = 1, ..., R$
     - *ii. Obtain* $TH_i^{C_k^{(r)}}$, $r = 1, ..., R$, *from the last iteration of Algorithm B.2*
   - *(b)* Downstream walk: *For stages* $i = 1, ..., Z-1$ *do*
     - *i. Use Algorithm B.2 to obtain the service rates of the synchronization stations* $J_{k,i-1}^{(r)}$ *and* $J_{b,i}^{(r)}$, $r = 1, ..., R$
     - *ii. Obtain* $TH_i^{C_b^{(r)}}$, $r = 1, ..., R$, *from the last iteration of Algorithm B.2*
3. *Analyze stage* i, $i = 1, ..., Z$, *in isolation to obtain its relevant performance measures*
4. *Analyze synchronization station* $J_{k,Z}^{(r)}$, $r = 1, ..., R$, *in isolation to obtain the market service levels*

Algorithm B.2 *Iteration to obtain the service rates* $\mu^{(r)}_{J_{k,i-1}}(1)$ *and* $\mu^{(r)}_{J_{b,i}}(1)$ *in stage* i

1. *Set the service rates* $\mu^{(r)}_{J_{k,i-1}}(1)$ *and* $\mu^{(r)}_{J_{b,i}}(1)$, $r = 1, ..., R$, *to some initial value*
2. *Repeat steps* (a) *and* (b) *until convergence of the service rates*
   (a) *Analyze the complement network of each single synchronization station* $J^{(r)}_{k,i-1}$ *and* $J^{(r)}_{b,i}$, $r = 1, ..., R$, *and obtain its throughput,* $TH_i^{C_k^{(r)}}$ *and* $TH_i^{C_b^{(r)}}$ *using algorithm B.3*
   (b) *Reset the service rates* $\mu^{(r)}_{J_{k,i-1}}(1)$ *and* $\mu^{(r)}_{J_{b,i}}(1)$ *of all synchronization stations* $J^{(r)}_{k,i-1}$ *and* $J^{(r)}_{b,i}$, $r = 1, ..., R$, *based on its external arrival process and the* $TH_i^{C_k^{(r)}}$ *and* $TH_i^{C_b^{(r)}}$ *obtained in step* (a)

Algorithm B.3 *Multi Class Approximate Mean Value Analysis with synchronization stations*

1. (Initialization) *For* $j = 0, \ldots, M$

$$\begin{aligned} ES_j(\overrightarrow{0}) &= \sum_{r=1}^{R} ES_j^{(r)} \\ ES_j^2(\overrightarrow{0}) &= \sum_{r=1}^{R} E\left(S_j^{(r)}\right)^2 \end{aligned}$$

if $c_j > 1$

$$\begin{aligned} p_j(0|\overrightarrow{0}) &= 1 \\ Q(\overrightarrow{0}) &= 0 \\ EL_{Q,j}(\overrightarrow{0}) &= 0 \end{aligned}$$

if $c_j = 1$

$$\begin{aligned} EL_j(\overrightarrow{0}) &= 0 \\ TH_j(\overrightarrow{0}) &= 0 \end{aligned}$$

For $r = 1, \ldots, R$

$$TH_j^{(r)}(\overrightarrow{0}) = 0$$

2. *For* $n = 1$ *to* N, *for all states* $\overrightarrow{n}$ *for which* $\sum_{r=1}^{R} n^r = n$ *and* $0 \leq n^r \leq N^r$

   (a) *For* $r = 1, \ldots, R$ *and* $j = 0, \ldots, M$, *if* $c_j > 1$

$$\begin{aligned} ES_{rem,j}^{(r)}(\overrightarrow{n}) &= \frac{c_j - 1}{c_j + 1} \frac{ES_j(\overrightarrow{n} - \overrightarrow{e_r})}{c_j} + \\ &\quad \frac{2}{c_j + 1} \frac{1}{c_j} \frac{ES_i^2(\overrightarrow{n} - \overrightarrow{e_r})}{2ES_j(\overrightarrow{n} - \overrightarrow{e_r})} \\ EW_j^{(r)}(\overrightarrow{n}) &= Q_j(\overrightarrow{n} - \overrightarrow{e_r}) ES_{rem,j}^{(r)}(\overrightarrow{n}) + \\ &\quad EL_{Q,j}(\overrightarrow{n} - \overrightarrow{e_r}) \frac{ES_j(\overrightarrow{n} - \overrightarrow{e_r})}{c_j} + ES_j^{(r)} \end{aligned}$$

   if $c_j = 1$

$$\begin{aligned} EW_j^{(r)}(\overrightarrow{n}) &= \sum_{s=1}^{R} \left[EL_j^{(s)}(\overrightarrow{n} - \overrightarrow{e_r}) - TH_j^{(s)}(\overrightarrow{n} - \overrightarrow{e_r}) ES_j^{(s)} \right] ES_j^{(s)} + \\ &\quad TH_j(\overrightarrow{n} - \overrightarrow{e_r}) ES_j(\overrightarrow{n} - \overrightarrow{e_r}) \frac{ES_j^2(\overrightarrow{n} - \overrightarrow{e_r})}{2ES_j(\overrightarrow{n} - \overrightarrow{e_r})} + ES_j^{(r)} \end{aligned}$$

(b) For $r = 1, \ldots, R$

$$TH_0^{(r)}(\overrightarrow{n}) = \frac{n^r}{\sum_{i=0}^{M} V_i^{(r)} EW_i^{(r)}(\overrightarrow{n})}$$

For $j = 0, \ldots M$

$$TH_j^{(r)}(\overrightarrow{n}) = V_j^{(r)} TH_0^{(r)}(\overrightarrow{n})$$

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

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

if $TH_j(\overrightarrow{n}) > 0$

$$ES_j(\overrightarrow{n}) = \sum_{r=1}^{R} \frac{TH_j^{(r)}(\overrightarrow{n})}{TH_j(\overrightarrow{n})} ES_j^{(r)}$$

$$ES_j^2(\overrightarrow{n}) = \sum_{r=1}^{R} \frac{TH_j^{(r)}(\overrightarrow{n})}{TH_j(\overrightarrow{n})} E\left(S_j^{(r)}\right)^2$$

(d) For $j = 0, \ldots, M$, *if* $c_j > 1$ *and* $n < c_j$

$$Q_j(\overrightarrow{n}) = 0$$

if $c_j > 1$ *and* $n \geq c_j$

$$Q_j(\overrightarrow{n}) = \frac{ES_j(\overrightarrow{n})}{c_j} \sum_{r=1}^{R} TH_j^{(r)}(\overrightarrow{n}) \left[Q_j(\overrightarrow{n} - \overrightarrow{e_r}) + p_j(c_j - 1 | \overrightarrow{n} - \overrightarrow{e_r})\right]$$

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

$$\frac{k}{ES_j(\overrightarrow{n})} p_j(k|\overrightarrow{n}) = \sum_{r=1}^{R} TH_j^{(r)}(\overrightarrow{n}) p_j(k|\overrightarrow{n} - \overrightarrow{e_r})$$

$$p_j(0|\overrightarrow{n}) = 1 - Q_j(\overrightarrow{n}) - \sum_{l=1}^{c_j - 1} p_j(l|\overrightarrow{n})$$

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

$$EL_{Q,j}(\overrightarrow{n}) = \frac{ES_j(\overrightarrow{n})}{c_j} \sum_{r=1}^{R} TH_j^{(r)}(\overrightarrow{n}) \left[EL_{Q,j}(\overrightarrow{n} - \overrightarrow{e_r}) + Q_j(\overrightarrow{n} - \overrightarrow{e_r})\right]$$

(g) For $r = 1, \ldots, R$ *and* $j = 0, \ldots, M$

$$EL_j^{(r)}(\overrightarrow{n}) = TH_j^{(r)}(\overrightarrow{n}) EW_j^{(r)}(\overrightarrow{n})$$

Appendix C

Market service levels

Fill rate

$$\begin{aligned} FR &= P(m_Z > 0) \\ &= \sum_{m_Z=1}^{S_Z} \pi(m_Z) \\ &= \pi(0) \sum_{m_Z=1}^{S_Z} \frac{\prod_{j=0}^{m_Z-1} TH_Z^{Cb}(S_Z - j)}{(\lambda_E)^{m_Z}}. \end{aligned} \tag{C.1}$$

Expected number of finished products

$$\begin{aligned} EL_{m,Z} &= \sum_{m_Z=0}^{S_Z} m_Z pi(m_Z) \\ &= \pi(0) \sum_{m_Z=1}^{S_Z} m_Z \frac{\prod_{j=0}^{m_Z-1} TH_Z^{Cb}(S_Z - j)}{(\lambda_E)^{m_Z}} \end{aligned} \tag{C.2}$$

Stockout probability If $N_Z = S_Z$

$$\begin{aligned} SP &= P(k_Z > 0) \\ &= \pi(0) \left(\frac{1}{1 - \frac{\lambda_E}{TH_Z^{Cb}(N_Z)}} - 1 \right) \end{aligned} \tag{C.3}$$

and if $N_Z > S_Z$

$$\begin{aligned} SP &= P(k_Z > 0) \\ &= \pi(0) \left[\sum_{k_Z=1}^{N_Z - S_Z - 1} \frac{(\lambda_E)^{k_Z}}{\prod_{j=1}^{k_Z} TH_Z^{Cb}(S_Z + j)} + \right. \\ &\quad \left. \frac{(\lambda_E)^{N_Z - S_Z}}{\prod_{j=1}^{N_Z - S_Z} TH_Z^{Cb}(S_Z + j)} \frac{1}{1 - \frac{\lambda_E}{TH_Z^{Cb}(N_Z)}} \right] \end{aligned} \tag{C.4}$$

Expected number of backordered demands If $N_Z = S_Z$

$$\begin{aligned} EL_{k,Z} &= \sum_{k_Z=0}^{\infty} k_Z \, \pi\left(-k_Z\right) \\ &= \pi\left(0\right) \frac{\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}}{\left(1-\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}\right)^2}. \end{aligned} \tag{C.5}$$

and if $N_Z > S_Z$

$$\begin{aligned} EL_{k,Z} &= \sum_{k_Z=0}^{\infty} k_Z \, \pi\left(-k_Z\right) \\ &= \pi\left(0\right) \sum_{k_Z=1}^{N_Z-S_Z-1} \frac{k_Z \, \left(\lambda_E\right)^{k_Z}}{\prod_{j=1}^{k_Z} TH_Z^{Cb}\left(S_Z+j\right)} + \\ &\quad \frac{\pi\left(S_Z-N_Z\right)}{\left(\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}\right)^{N_Z-S_Z}} \left[\frac{\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}}{\left(1-\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}\right)^2} - \right. \\ &\quad \left. \sum_{k_Z=0}^{N_Z-S_Z-1} k_Z \left(\frac{\lambda_E}{TH_Z^{Cb}\left(N_Z\right)}\right)^{k_Z} \right] \end{aligned} \tag{C.6}$$

Expected customer order response time

$$ECORT = \frac{EL_{k,Z}}{\lambda_E} \tag{C.7}$$

Expected waiting time of a backordered demand

$$EWBD = \frac{EL_{k,Z}}{\left(1-FR\right)\lambda_E} \tag{C.8}$$

In these equations, $\pi\left(0\right)$ is:
If $N_Z = S_Z$

$$\pi\left(0\right) = \left[\sum_{m_Z=1}^{S_Z} \frac{\prod_{j=0}^{m_Z-1} TH_Z^{Cb}\left(S_Z-j\right)}{\left(\lambda_E\right)^{m_Z}} + \frac{1}{1-\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}} \right]^{-1} \tag{C.9}$$

If $N_Z > S_Z$

$$\begin{aligned} \pi\left(0\right) = \left[1 + \sum_{m_Z=1}^{S_Z} \frac{\prod_{j=0}^{m_Z-1} TH_Z^{Cb}\left(S_Z-j\right)}{\left(\lambda_E\right)^{m_Z}} + \sum_{k_Z=1}^{N_Z-S_Z-1} \frac{\left(\lambda_E\right)^{k_Z}}{\prod_{j=1}^{k_Z} TH_Z^{Cb}\left(S_Z+j\right)} \right. \\ \left. + \frac{\left(\lambda_E\right)^{N_Z-S_Z}}{\prod_{j=1}^{N_Z-S_Z} TH_Z^{Cb}\left(S_Z+j\right)} \frac{1}{1-\frac{\lambda_E}{TH_Z^{Cb}(N_Z)}} \right]^{-1} \end{aligned} \tag{C.10}$$

Appendix D

Application extensions

The Decision Support System developed by van 't Klooster [Klo03] can be used for analyzing a manufacturing system. It deals with many disadvantages of existing applications. First, the application implements a better and more realistic model. Secondly, users can insert data, perform the analysis and view the results in a very intuitive way.
In this chapter, the implementation of some extensions to the model and the application is described. These extensions upgrade the DSS so it becomes a better support to decision makers in making more and more complex decisions.

The first extension is the modeling of operators. For that, the algorithm developed in the previous chapter is used. But this new algorithm has to be tested first and compared with the current used algorithm[1]. Therefore it must be possible to run several algorithms, and compare their results. This is the second extension.

D.1 Current application

First, in the new implementation all practical aspects given in [Klo03] can be included in the algorithm.
Secondly, the algorithm is optimized. The current algorithm contains an upstream and downstream walk. But these can only be defined properly when the system is a line system and all products visit the cells according to this line. This upstream and downstream walks are not necessary for convergence. Any order of analyzing the cells will work. However, an efficient order can be a big reduction of the computational effort. But this efficient order depends on the routing of the products, and it is difficult to give a general procedure for it. Therefore we choose the most easy implementation: analyze every cell once in the order the cells are created. This can be repeated until convergence.

Also a different strategy in Algorithm B.2 is possible. In Algorithm B.2 the service rates are reset only after analyzing the complement network of each synchronization station. Instead the service rates can be reset after only analyzing its own complement network (yielding new values for the arrival rate of Kanbans at the station). This strategy can lead to a faster convergence of the algorithm and thus reduce its computational effort.

[1] Algorithm 1 can also be used when no operators are modeled

D.2 Implementation of extensions

To test any new algorithm, it is best to use the input possibilities of the current application. This will speed up the testing procedure, because one does not have to worry about specifying the input and obtaining the output in a nice way. Only the actual algorithm has to be programmed. The idea is that it will be very easy to add a new algorithm and view the results of it. To do so, a layout for an algorithm is defined. In this layout, the reading and writing of data is defined, the new algorithm only have to be implemented in a specified procedure. Only code has to be written which uses this layout to compute performance measures.
There are already several algorithms present in new application. Of course the current algorithm for a MC GKCS with an AMVA procedure with or without *partial reduction*. These two AMVA procedures can also be used for a CQN, and thus appear in the list with existing algorithms. Also the new algorithm developed in chapter 3 is already implemented.

The adding of operators is very straightforward. Only an extra object, with some fields, has to be added to the object structure. The most important fields are the number of operators, and the cell which contains the operator group.
Also, some existing objects need a field to hold a reference to an operator group.

Because the new algorithm for the operators is used for an open queueing network first, the current object structure of a manufacturing system has to be extended, such it can handle other network types, not only kanban systems. There are four types of networks defined: *kanban systems*, *Closed Queueing Network* (CQN), *Semi Open Queueing Network* (SOQN) and *Open Queueing Network* (OQN). A parameter is added which stores the type of network. This is a property of the factory, and has to be given by the user (via *factory properties*). Because of generality and official purposes of the application, the default value is set to *kanban*.
Based on this field, some options and possibilities can be disabled or invisible in the application, because they are not relevant for the chosen network type. For instance, the kanban system is the only one that can handle multiple stages in a network. Therefore, when the type is one of the others, only one cell can be defined.

The difference for the CQN, SOQN and the OQN is the role of the synchronization stations. For all of this types, there is no arrival of demands (no base stock) at the last synchronization station. This can be obtained by setting the arrival rate of demands to infinity. For a CQN, there is no arrival of products at the beginning of the network. This means the arrival rate at the first synchronization station have to be set to infinity. An OQN does not have any pallets circulating around. This can be obtained by setting the service rate of the first synchronization station to infinity.

To make it possible to show the results for different algorithms at the same time, scenarios are implemented. Before the algorithm started, first one or more scenarios have to be chosen. The results of all these scenarios are shown in the results.
A scenario contains the algorithm it uses. But it can also be used to show result from another database. This is useful when the results for one algorithm or situation are known, and these have to be compared with the results of another scenario. In this way, they are shown in the same figure, but the first situation does not have to be ran anymore.

Another use of scenarios is to see the effect on the performance measures when some input parameters change. The possibility to define these changes is not implemented fully, so this option is not possible yet.

Finally, some things change in the checks that are performed at the data before the calculation starts. The data error *Incomplete flow over cells* cannot occur anymore. Thus this does not have to be checked anymore.
An extra error is added when the selected algorithm does not fit the selected network type. For instance, AMVA cannot be used when the network is a kanban system.

The next table gives the extra fields in the current object structure of the application, due to all the aspects mentioned in this chapter.

Object	**Fields**	**Store?**
Factory	Type of network (OQN, SOQN, CQN, Kanban)	Y
	Are operators present?	Y
	List with all existing operator groups	NO
	List with all existing scenarios	NO
Cell	List with all the operator groups in the cell	NO
	Type of network (OQN, SOQN, CQN, Kanban)	NC
Operator group	Unique ID	Y
	Name	Y
	Number of operators in the group	Y
	Cell which contains the operator group	Y
	List with every operation which need this operator group	NO
Operation	When the operator is needed	Y
	Operator group which is used	Y
Scenario	Unique ID	Y
	Name	Y
	Used algorithm	Y
	Database name	Y
	Changes in the data	Y

Not all of these fields have to be stored in the database, some can be computed when their value is necessary. To store them is just a waste of memory. Which fields have to be stored is also given in the last column of the table. *Y* means that the field returns in the table of the corresponding object, *NO* means that the data is stored, but not as a field of the corresponding object. Mostly, these are lists which are stored in another table. *NC* means that the value can be computed from values of other fields. Thus this field is not stored in the database.
For the operator group, a table is added to the database, to store its fields.

D.3 Graphical User Interface

All these extensions also have their effect on the Graphical User Interface. Windows and menu options have been added or changed.
The first thing that have to be done now for a new factory is setting its properties (*View*

→ *Factory Properties*) This screen is shown in figure D.1

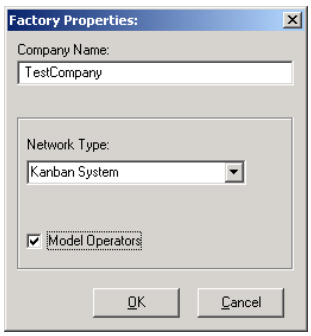

Figure D.1: Factory Properties Window

The most important are the network type and if operators are used in the factory. The values of these properties will affect the look and possibilities of other windows. The *Company Name* is used as a header when reports are printed.

Adding an operator group to a cell works exactly the same as adding a resource to a cell. For every cell a list with operator group can be opened, in which operator groups can be added, edited and deleted.

When all the factory data is inserted, scenarios can be made (*View* → *Scenarios*). This window is shown in figure D.2.

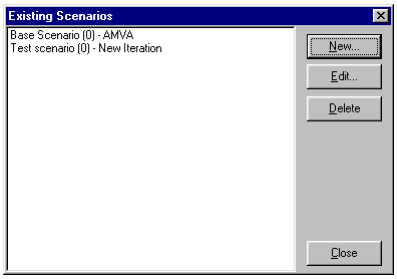

Figure D.2: Existing Scenarios Window

There always exists one scenario, *Base Scenario*. This is the standard scenario used in calculation. Initially it uses the default algorithm, but this can be changed.

The details of a scenarios can be edited in a seperate window (figure D.3).
When a list with available scenarios is shown somewhere in the application, the description of a scenario consists of three part: the name of the scenario, the number of changes in the data between brackets and the algorithm that is used in the scenario.

When also scenarios are defined, the algorithm can be run. This process consists of several steps, which are put into a wizard (*Run* → *Calculate*), to help the user.
In the first step of the running (figure D.4), one has to select one or more scenarios. In the

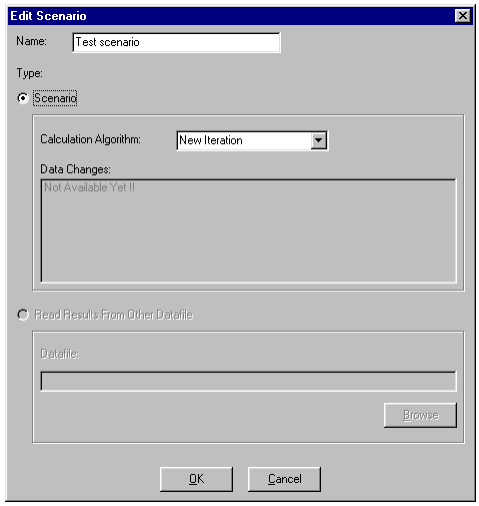

Figure D.3: Edit Scenarios Window

next step, the data is checked. When errors or warnings are found, these are shown (figure D.5). when there are errors the running is stopped. With only warnings, one can continue. When no errors or warnings are found, this window is skipped.
Next a warning is shown, that the running can take a long time when a lot of data is present. Finally, when the running is finished, this is given in the last screen (figure D.6), together with the time the run took. This is for algorithm-testing purposes. Here is also shown if parts of the system are unstable, non-converging or if an unexpected error has occurred.

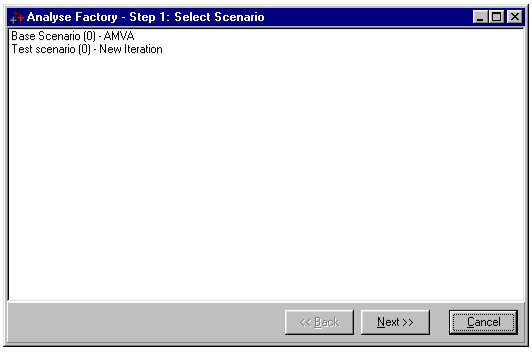

Figure D.4: First step of the wizard

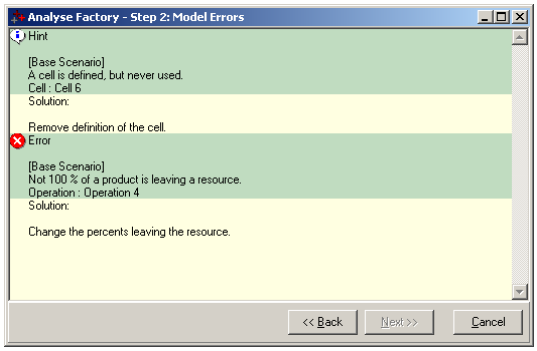

Figure D.5: Second step of the wizard

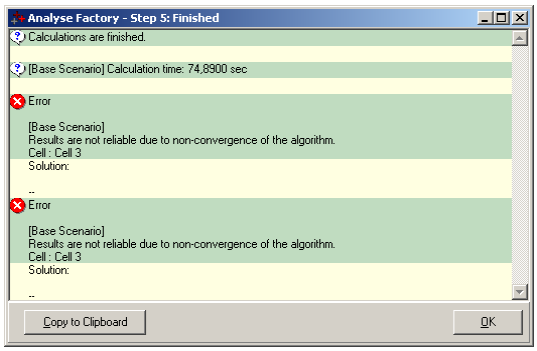

Figure D.6: Last step of the wizard

D.4 Further improvements

Several things need to be done in the application. These are not fully implemented parts. An important one is to make it possible to run different scenarios with changes in the factory data.

When this running of scenarios is fully implemented, optimization methods are easy to implement. An optimization method only is analyzing the network for several parameter sets, and pick the best set. When scenarios are used, an optimization method only has to define several scenarios, and these can be used to analyze the network. Then the best solution can be picked by the user, or by the optimization method itself, if the *best* parameter set is well-enough defined.