Control of Manufacturing Systems using Effective Process Times

S.A. Coenen

SE 420439

Master's Thesis

Supervisor: Prof.dr.ir. J.E. Rooda
Coach: Dr.ir. A.A.J. Lefeber

Eindhoven University of Technology
Department of Mechanical Engineering
Systems Engineering Group

Eindhoven, June 2005

ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY June 2005
Department of Mechanical Engineering
Systems Engineering Group

Student	S.A. Coenen
Supervisor	Prof.dr.ir. J.E. Rooda
Advisor	Dr.ir. A.A.J. Lefeber
Start	June 2004
Finish	June 2005
Title	Control of Manufacturing Systems using Effective Process Times

Subject

Complex manufacturing systems, with workstations at high utilization levels, can be controlled using different techniques. Scheduling is such a technique, requiring a stable and predictable manufacturing environment. Unfortunately this prerequisite is rarely met in a complex manufacturing environment, since variability has a corrupting influence.

In the past, research has been performed how to control such complex manufacturing systems in order to maximize throughput and to reduce flow time. Hierarchical Model Predictive Control frameworks have been developed using several control layers. Research has led to the belief that information of the aggregated state in the form of Effective Process Times can be used to perform the planning in the high level control layer. Model Predictive Control provides feasible short-term production targets for the low level control layer, which consists of a flow rate controller and sequencing policy. It uses the received production targets and the current state of the system to make short-term decisions in order to achieve the targets.

Assignment

Previous work has shown that the implemented control framework does not always function properly when applied to a practical case. Investigate by using small cases why the high level control layer and in particular the optimization tool, fails.

Furthermore, when the controlled system is subject to condition blocking, the EPT measurements are not accurate with the consequence that the characteristic curve is not a correct representation of the system. Therefore, when delivering targets to the low level control layer, unrealistic low targets are set. They will cause a lower utilization of the system and the EPTs will again grow. Investigate how this vicious circle can be broken, either by developing a new control strategy or by reviewing the existing EPT algorithms.

Determine how different control strategies, implemented in the hierarchical Model Predictive Control framework, lead to several solutions by looking at the performance measures of the manufacturing system. Present an extensive discussion of the results in a report and conclude with recommendations for further research.

Prof.dr.ir. J.E. Rooda Dr.ir. A.A.J. Lefeber

Preface

What lies before you is the result of the work on my final assignment for becoming a Master of Science. This thesis finishes a five year curriculum at the Department of Mechanical Engineering of the Eindhoven University of Technology. The final years I spent with the Systems Engineering Group. The choice for this group was pretty obvious for me; the second year course 'Modeling of Industrial Systems' and the overall research theme of this group appealed very much to me.

Fortunately, I was wise enough to explore all facets of the student life during my study. This includes living with many other student housemates, visiting many get-togethers and parties, undertaking study related activities and most importantly joining the rowing club 'E.S.R. Thêta'. My presidential activities and committee memberships, my modest experiences in race rowing and the contact with many fellow-members have been a valuable experience. All the above things also made sure I stretched the five year curriculum with a couple of years, but for my benefit, I am convinced about that.

I would like to thank professor Rooda for supervising this research project and for providing the necessary means to make sure my stay at the Systems Engineering group was a pleasant one. A great deal of thanks goes out to Erjen Lefeber, for his constructive support and pleasant cooperation during both my internship at Eldim BV and my Master's project. Also the help of many other staff members is much appreciated.

Some other thanks go out to all fellow-students and PhD students from the SE-lab. Many of them pleasantly distracted me from my work with non-study related discussions and activities, like the mini-golf competition. Also thanks to my dear friends which I have since 'Gymnasium 2', you have always been there for me.

Last, but certainly not least, I would like to thank my family. My late father, who has proved to be a strong and positive minded man putting up with the big C. I am very grateful to you dad and hope you are proud of all my achievements. My mother, who did not limit me in my activities, although it was sometimes hard to live without her husband and with both sons living in Eindhoven. I think you have proved to be a much stronger person than you yourself could ever imagine. Finally, my younger brother Rob, for all brotherhood related activities and distractions. I am glad you finally decided to grow up ;-).

Bas Coenen – S.A.Coenen@gmail.com

Summary

Due to the rapidly increasing complexity of manufacturing systems, heuristic methods become incapable of finding an optimal control strategy. Furthermore, heuristics cannot react on the dynamics of a manufacturing system, especially with workstations at high utilization levels, since variability has a corrupting influence. Therefore, in this research, the two layer hierarchical model predictive control framework, is considered, which uses effective process times as input parameters for the controller.

The hierarchical MPC framework has already been deployed in a simulation framework by Tolboom [Tol04]. However the control framework did not completely function as desired. Especially the optimization tool in the high level controller did not work properly. Also, the measurement of effective process times was incorrect due to the occurrence of condition blocking. The measured EPTs are transformed into the characteristic curve of each workstation of the system, which captures the highly non-linear relationship between throughput and work in process. The characteristic curves are used as capacity constraints for the optimization tool. Due to condition blocking, too high EPTs are measured, which causes the characteristic curves not to be a correct representation of the system. Therefore unrealistic low targets are set for the manufacturing system, since the controller has a too low estimate of the system's capacity. This causes a lower utilization of the system and the measured EPT realizations will again increase. This reciprocity causes a vicious circle.

The first objective of this research is to improve the proposed two layer model predictive control framework by 'refurnishing' the layers, with an emphasis on the high level optimization tool. Secondly, the effective process time measurement in case of condition blocking has be reviewed, so it can be correctly implemented into the framework.

The control framework consists of two layers; a high and low level controller. The high level control layer uses planning approaches to derive a set of feasible production targets based on the actual demand and the aggregated state of the system. It consists of a model predictive controller which optimizes a cost function over a future horizon, subject to a set of both linear and non-linear constraints. The linear constraints can be interpreted as mass conservation laws, since they capture the wip level behavior of the workstations. The non-linear constraints are represented by the characteristic curves of the workstations. Important parameters of the characteristic curve are obtained from EPT calculations.

EPTs are mainly used as a performance measure, but in this research, they are mostly used as input parameters for the model predictive controller. Several EPT algorithms exist, all applicable for different situations, like finitely or infinitely buffered workstations, blocking or not blocking, single or multiple lot machines. But until now, no suitable EPT algorithm is derived which can handle condition blocking. This type of blocking occurs often at the presented control framework, it means that a task has some condition on its input port, demanding this condition is fulfilled before activating the task. In this case more specifically; the controller sometimes intentionally leaves a machine idle, while it is empty and lots are waiting in the corresponding buffer.

Low level control consists of sequencing policies and flow rate control. Only the latter is implemented in the control framework in this research. The flow rate controllers are present in the buffers of the workstations and make sure the influx of the machines is each period equally distributed.

Evaluation of the control framework is performed with the use of a simple manufacturing line with two infinitely buffered machines. Several input and tuning parameters exist. The input parameter is declared as the demand type for the manufacturing system, which can be altered throughout the experiments. The tuning parameters are needed for a proper functioning of the optimization toolbox *Tomlab* and are determined by several preliminary test runs. The chosen performance measures are the buffer (or wip) levels of the workstations, the number of cumulative backorders and the effective process times.

In order to meet the first research objective, the derived control framework is tested. It is proved that the framework itself works properly, but the EPT measurement is wrong because it is dependent on the capacity utilization. Therefore, the control framework performs poorly for high utilization levels. Also the previously discussed vicious circle is detected for systems with extremely low utilization levels.

The second research objective is met by deriving a new EPT algorithm, which makes use of the authorization times which are determined by a combination of the high and low level controller. Now, the EPTs are no longer dependent on the utilization of the workstations. The control framework performs significantly better with the new EPT measurement for several types of demand and in both highly and extremely lowly utilized manufacturing systems than with the old EPT measurement. Also the described vicious circle is neutralized by this new method.

Samenvatting (in Dutch)

Doordat de complexiteit van fabricagesystemen blijft toenemen, blijken heuristische methoden incapabel om optimale regelstrategieën te vinden. Daarnaast kunnen heuristieken niet op het dynamische gedrag van een fabricagesysteem reageren. Dit gaat vooral op voor werkstations met hoge utilisatie niveaus, omdat variabiliteit een vervelende invloed heeft. Daarom wordt er in dit onderzoek een twee-laags hiërarchisch model predictive control raamwerk beschouwd, dat effectieve proces tijden gebruikt als ingangsvariabelen voor de regelaar.

Het hiërarchische MPC raamwerk is al eerder in een simulatieraamwerk geïmplementeerd door Tolboom [Tol04]. Echter werkte dit regelraamwerk niet zoals gewenst. Vooral de optimalisatie tool in de bovenste laag van het regelraamwerk werkte niet goed. Daarnaast bleek de meting van effectieve proces tijden incorrect door de aanwezigheid van condition blocking. De gemeten EPTs worden getransformeerd in de karakteristieke curve van ieder werkstation van het systeem, deze curve geeft de hoog niet-lineaire eigenschap tussen doorzet en onderhanden werk weer. De karakteristieke curves worden gebruikt als capaciteitsconstraints voor de optimalisatie tool. Door de aanwezigheid van condition blocking worden te hoge EPTs gemeten, wat er voor zorgt dat de karakteristieke curves geen correcte representatie van het systeem vormen. Daardoor worden onrealistisch lage doelen gesteld voor het fabricagesysteem, omdat de controller een te lage schatting van de systeemcapaciteit ontvangt. Dit zorgt voor een lagere utilisatie van het systeem en de gemeten EPT realisaties zullen opnieuw toenemen. Deze reciprociteit veroorzaakt een vicieuze cirkel.

Het eerste onderzoeksdoel is om het voorgestelde twee-laags model predictive control raamwerk te verbeteren door de lagen opnieuw te 'meubileren', waarbij de nadruk op de optimalisatie tool van de bovenste laag ligt. Op de tweede plaats moet de effectieve proces tijd meting in het geval van condition blocking worden herzien, zodat deze correct in het raamwerk kan worden geïmplementeerd.

Het regelraamwerk bestaat uit twee lagen; beide hebben hun eigen regelaar. De bovenste laag van het regelraamwerk gebruikt planningsmethoden om een set van feasible productiedoelen af te leiden, gebaseerd op de actuele vraag en de algehele toestand van het systeem. De laag bestaat uit een model predictive controller die een kostenfunctie minimaliseert over een toekomstige horizon, onderworpen aan een set van zowel lineaire

als niet-lineaire constraints. De lineaire constraints mogen als massabehoudswetten worden beschouwd, omdat zij het wip niveau gedrag van de werkstations beschrijven. De niet-lineaire constraints worden vertegenwoordigd door de karakteristieke curves van de werkstations. Belangrijke parameters voor de karakteristieke curve worden verkregen door EPT berekeningen.

EPTs worden meestal als prestatie-indicatoren gebruikt, maar in dit onderzoek vormen ze voornamelijk de ingangsvariabelen voor de model predictive controller. Verscheidene EPT algoritmes bestaan, welke allemaal op andere situaties toepasbaar zijn, zoals bij eindig en oneindig gebufferde werkstations, wel of geen blocking, één of meerdere product machines. Maar tot nu toe is er geen geschikt EPT algoritme ontwikkeld dat met condition blocking om kan gaan. Dit type blocking vindt vaak plaats in het geïntroduceerde regelraamwerk, en het houdt in dat een taak een bepaalde conditie op zijn inkomende poort heeft, die eist dat aan deze conditie is voldaan voordat de taak wordt geactiveerd. Meer specifiek voor dit geval betekent het dat de regelaar soms bewust een machine inactief laat, terwijl zij leeg is en producten in de bijbehorende buffer aan het wachten zijn.

De onderste laag van het regelraamwerk bestaat uit sorterings algoritmes en regelaars voor de toevoersnelheid. Alleen deze laatste soort is geïmplementeerd in het regelraamwerk van dit onderzoek. De regelaars voor de toevoersnelheid bevinden zich in de buffers van de werkstations en zorgen ervoor dat de toevoer van de machines iedere periode gelijk is verdeeld.

Het regelraamwerk wordt geëvalueerd met behulp van een simpele fabricage lijn, die twee oneindig gebufferde werkstations bevat. Verschillende ingangsvariabelen en instelparameters zijn van toepassing. Als ingangsvariabele wordt het type vraag voor het fabricage systeem genomen, die gedurende de experimenten wordt gevarieerd. De instelparameters zijn nodig voor een goede werking van de optimalisatie tool *Tomlab* en worden door middel van enkele test series vastgesteld. De gekozen prestatie-indicatoren zijn de buffer (of wip) niveaus, het aantal cumulatieve nabestellingen en de effectieve proces tijden.

Om aan het eerste onderzoeksdoel te voldoen, wordt het ontwikkelde regelraamwerk getest. Het is bewezen dat het regelraamwerk in principe werkt, echter de EPT meting is foutief, omdat deze van de capaciteits utilisatie afhankelijk blijkt. Daarom presteert het regelraamwerk slecht voor hoge utilisatie niveaus. Ook de eerder genoemde vicieuze cirkel is waargenomen voor systemen met zeer lage utilisatie niveaus.

Het tweede onderzoeksdoel is gehaald door het ontwikkelen van een nieuw EPT algoritme, dat gebruikt maakt van de autorisatietijden, die door een combinatie van de bovenste en onderste laag regelaars worden afgeleid. Nu zijn de EPTs niet meer afhankelijk van de utilisatie van de werkstations. Het regelraamwerk presteert duidelijk beter met de nieuwe EPT meetmethode voor verscheidene vraag types in zowel hoog als zeer laag geütilizeerde fabricagesystemen, dan met de oude EPT meetmethode. Ook de beschreven vicieuze cirkel wordt doorbroken door middel van deze nieuwe methode.

Contents

List of Definitions

Actual arrival	The moment that lot i physically arrives on workstation j.
Actual departure	The moment that lot i physically departs from workstation j.
Blocking	The situation where a workstation, upon finishing processing of a lot, cannot send the lot on because the next workstation cannot receive the lot.
Characteristic curve	Expression for the expected output of (a part of) the manufacturing system as a function of the WIP over a given period of time.
Condition blocking	All types of blocking that occur since a certain condition on an input port is not (yet) met.
Dispatched work	Set of lots (work) released into the manufacturing system.
Dispatching	Determine which lot to process next, at the moment a resource becomes idle.
Downstream workstation	Receiving workstation (after another), i.e. the most downstream workstation is the last in the manufacturing line.
Effective process time	The total time seen (claimed / consumed) by a lot on a workstation.
EPT realization	The time a lot was in process plus the time a lot (not necessarily the same lot) could have been in process.
Port blocking	Port blocking occurs when both the sending and receiving resource are ready to, respectively, send or receive a lot, but the lot remains on the sending resource for some reason.
Possible arrival	The moment that lot i could have arrived on workstation j, i.e. that workstation $j-1$ finished processing of the lot.
Possible departure	The moment that workstation j finishes processing of lot i and tries to send the lot.
Possible work	Set of lots (work) that are allowed to be processed, with respect to the issued production targets.
Resource blocking	Blocking caused by the finiteness of the buffers in the system.
Scheduling	Allocation of a specified lot process action to a resource at a specified time segment.

Sequencing	Sorting of the available work with respect to a predefined algorithm.
Server	Single lot machine.
Supervisory controller	High level controller.
Unit	Low level controller.
Upstream workstation	Sending workstation (in front of another), i.e. the most upstream workstation is the first in the manufacturing line.
Workstation	A combination of one or more parallel machines and zero or more buffer spaces.

List of Acronyms and Symbols

Acronyms

1SLM	One Single Lot Machine
1SLMB	One Single Lot Machine, subject to Blocking
CRP	Capacity Requirements Planning
DEM	Discrete Event Model
DES	Discrete Event System
EDD	Earliest Due Date
EPT	Effective Process Time
ERP	Enterprize Resources Planning
FIFO	First-In First-Out
JIT	Just-In-Time production
LS	Least Slack
MPC	Model Predictive Control
MRP-I	Material Requirements Planning
MRP-II	Manufacturing Resources Planning
mSLM	Multiple Single Lot Machine
mVBM-r	Multiple Batch Machines with Variable batchsizes and processing multiple Recipes
OEE	Overall Equipment Efficiency
RHS	Receding Horizon Strategy / Rolling Horizon Scheme
SLM	Single Lot Machine
SPT	Shortest Process Time
SRPT	Shortest Remaining Process Time
WIP	Work In Process

Symbols

δ	Throughput
φ	Cycle time
φ_q	Cycle time of the queue

τ	Time of occurrence
$\tau^{f}_{i,j}$ or $\tau^{f_e}_{i,j}$	Finish time of the EPT realization of lot i on machine j
$\tau^{f_p}_{i,j}$	Finish time of the PB realization of lot i on machine j
$\tau^{s}_{i,j}$ or $\tau^{s_e}_{i,j}$	Start time of the EPT realization of lot i on machine j
$\tau^{s_p}_{i,j}$	Start time of the PB realization of lot i on machine j
$\mathbf{AA}_{i,j,k}$	Actual arrival (i.e. the moment that lot i physically arrives at machine j of workstation k)
$\mathbf{AT}_{i,j}$	Authorization time of lot i at workstation j
$\mathbf{AD}_{i,j,k}$	Actual departure (i.e. the moment that lot i physically departs from machine j of workstation k)
$\mathbf{BE}_{i,j}$	The time that the buffer of workstation $j+1$ last changed from 'saturated' to 'one buffer space available'
$\underline{c}$	Cost vector
c_a^2	Squared coefficient of variation of the inter arrival time
c_e^2	Squared coefficient of variation of the effective process time
$d(t)$	Realized demand in period $]t, t+1[$
$d(t+i\|t)$	At time t expectation of the demand for period $]t+i, t+i+1[$
$EPT_{i,j}$	EPT realization of lot i at workstation j
$EPT_{i,j,k}$	EPT realization of lot i at machine j of workstation k
$\mathbf{PA}_{i,j}$	Possible arrival (i.e. the moment that lot i could have arrived at workstation j)
$\mathbf{PD}_{i,j}$	Possible departure (i.e. the moment that lot i could have departed from workstation j)
t_e	Mean effective process time
u	Utilization
x	Throughput
$x_k(t)$	Realized production at workstation k in period $]t, t+1[$
$x_k(t+i\|t)$	At time t planned target for workstation k for period $]t+i, t+i+1[$
$y(t+i\|t)$	At time t planned surplus in stock for time $t+i$, assuming that the planning is perfectly executed
w	Work in process
$w_k(t)$	Measured wip of workstation k at time $t^{(-)}$
$w_k(t+i\|t)$	At time t expectation of the wip of workstation k for time $t+i$, assuming that the planning is perfectly executed
$z(t+i\|t)$	At time t planned backorders for time $t+i$, assuming that the planning is perfectly executed

Chapter 1

Introduction

> *"Scientists study the world that is; engineers create the world that never was."*
> - Theodore von Karman (1881-1963)

Obviously, Von Karman was an engineer, in fact a mechanical engineer. His statement is a daring one, but has a grain of truth in it. Engineering has been described as the art of the practical application of scientific principles to 'directing the great sources of power in nature for the use and convenience of man'. It involves men, money, material, machine and energy and 'requires above all the creative imagination to innovate useful applications of natural phenomena'. It also has the character of a never-ending search for 'newer, cheaper, better means of using natural sources of energy and materials to improve man's standard of living and to diminish labor'.

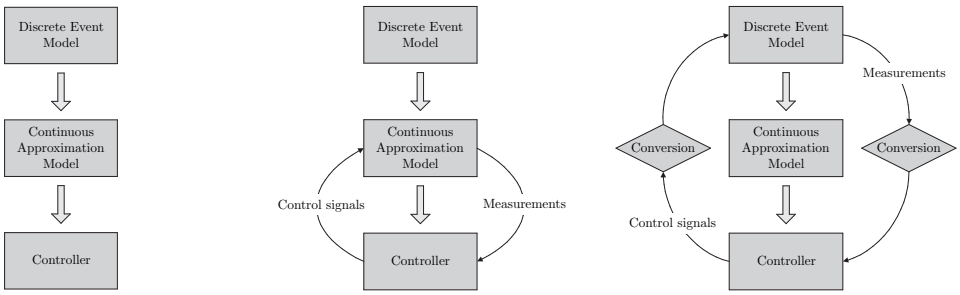
The Systems Engineering Group aims to develop methods, techniques and tools for the design of advanced industrial systems. By using scientific principles from mechanical engineering, computer science and mathematics, systems engineers try to create a new, controllable manufacturing world. The adjective 'controllable' is a very important part of this sentence. Engineers always try to control their environment and the field of manufacturing is absolutely no exception to that.

Control of manufacturing systems can be performed at different levels and its purpose is to satisfy the engineer demands in an 'optimal' way. Here, the definition of 'optimal' is system dependent and can therefore be read in various ways, such as 'with the least possible cost' or 'in the shortest possible time'.

In the past, many heuristic methods have been developed for the control of a manufacturing system, such as material requirements planning (MRP), enterprize resources planning (ERP) or just-in-time production (JIT). Nowadays, many heuristic methods are still being used in combination with operator experience for management of resources and planning of production. However, as the complexity of the manufacturing

system increases rapidly, the heuristic methods and operator experience will at some point become incapable of finding an optimal control strategy. Furthermore, heuristics can not react on the dynamics of a manufacturing system. Therefore, in this research, a different control framework will be studied, which is dynamical.

It is not possible to apply this dynamic control theory directly to a manufacturing system. The problem preventing this, is that most control theory is meant for continuous systems, whereas the considered system is not of a continuous nature. In this research, a discrete event model is developed, which is supposed to be equal to the physical manufacturing system and is represented as a χ-model (for more information about the specification language χ, see Kleijn [Kle01]). In order to be able to apply control theory, the discrete event model is approximated by a continuous model, see Figure 1.1a. Using feedback control techniques, a controller is designed for this approximation model, see Figure 1.1b. Once a good controller has been developed, it is connected to the discrete event system. Hereby, the output of the controller should be converted in such a way that it becomes a suitable input for the discrete event system, and vice versa, see Figure 1.1c.



(a) Step 1: Continuous approximation of the discrete event model.

(b) Step 2: Control of the continuous approximation model.

(c) Step 3: Conversion is used to connect the controller to the discrete event model.

Figure 1.1: The dynamic control framework.

This research focuses on the development of the controller. As previously stated, a controller only consisting of heuristics is not suitable to control complex manufacturing systems, with workstations at high utilization levels, since variability has a corrupting influence.

In the recent past hierarchical model predictive control frameworks have been developed [Var03], using several control layers. Research has led to the belief that information of the aggregated state can be used as an input parameter to perform the planning in the high level control layer by means of an optimization tool. The optimization provides feasible short-term production targets for the low level control layer, which consists of a flow rate controller and sequencing policy. The low level control layer uses the received production targets and the current state of the system to make short-term decisions in order to achieve the targets.

The input parameters for the high level layer can be measured in the form of effective process times. The introduction of EPT as a concept to account for throughput losses and process time irregularities on workstations has been made by Hopp and Spearman [Hop01]. Further investigation as a means of system analysis has been performed by Jacobs *et al.* [Jac01, Jac03], Van Bakel [Bak01], Rooney [Roo02], Wullems [Wul02] and Kock [Koc03].

1.1 Objective

Tolboom [Tol04] has implemented a two layer version of the hierarchical model predictive control framework of Vargas-Villamil *et al.* [Var03] in the Intel Case [Kem03]. However the control framework did not completely function as desired. Especially the optimization tool in the high level control layer did not work properly.
Also, an input parameter of the optimization problem is not correct. Effective process time measurements of the system under control are used as capacity constraints for the optimization tool. These EPTs are therefore transformed into the so-called characteristic curve, which captures the highly non-linear relationship between throughput and work in process. At the moment, when the controlled system is subject to condition blocking, the measured EPT realizations are too high with the consequence that the characteristic curve is not a correct representation of the system. Therefore unrealistic low targets are set, when delivering targets to the low level control layer, since the controller has a too low estimate of the system's capacity. This causes a lower utilization of the system and the measured EPT realizations will again increase. This reciprocity causes a vicious circle.

The goal of this research is twofold. First, the proposed two layer model predictive control framework has to be improved. Secondly, the precise definition of effective process time and the algorithms to compute it have to be reviewed, so they can be implemented into the hierarchical MPC framework. Validation of the adapted controller must be performed by implementing it in several test cases.

1.2 Approach and Outline

In order to meet these objectives, the following approach is used, which also defines the outline of this report.

Chapter 2 contains a literature review on manufacturing control approaches. First, three different classifications of control frameworks are mentioned; hierarchical, heterarchical and hybrid. This research makes use of a two layer hierarchical framework. It is explained that model predictive control is chosen as the high level control layer, and that low level control is performed by a flow rate controller and sequencing policies.

Finally, a brief introduction is given on the effective process time algorithm, since it is used as a means of conversion in the control framework.

The effective process time algorithm has already been briefly introduced. Chapter 3 gives a more extensive survey on this matter by examining many forms of the EPT algorithm in literature. The main use of EPTs is as a performance measure, but in this research, EPT calculation is mostly used for determination of the characteristic curve, which is used as a non-linear constraint for the optimization process in the high level controller. Several EPT algorithms exist, all applicable for different situations, like blocking or single and multiple lot machines. In this chapter, the EPT algorithms encountered in literature, which are useful for this research, are discussed.

Chapter 4 handles all three parts of the proposed two layer hierarchical model predictive control framework in detail. First, the discrete event system is examined and a suitable model is proposed. Then the high level control layer is discussed, with an extensive description of the model predictive controller, which consists of an optimization algorithm. It minimizes a cost function, subject to non-linear capacity constraints and linear mass conservation laws. Finally the low level layer is described, which primarily examines the flow rate controllers, which are present in the buffers of the manufacturing system.

Chapter 5 evaluates the proposed framework in an experimental setup. It focusses on two main subjects, where the first is to validate the control framework and the second is to improve the EPT measurement in case of condition blocking. First, simulation assumptions are made and the demand type is chosen as input parameter of the model. The performance measures are determined to be the wip levels of the workstations, the number of cumulative backorders and the effective process times.
Experiments are conducted which prove that the proposed control framework functions properly, only the EPT measurements turn out to be dependent on the capacity utilization, which is wrong per definition. Therefore a new EPT algorithm is derived, which makes use of the authorization times which are determined by a combination of the high and low level controller. Now, the EPTs are no longer dependent on the utilization of the workstations. The control framework functions much better for several types of demand and levels of utilization with this new algorithm, which is proved by conducting several experiments.

Finally, in Chapter 6 the conclusions and recommendations for further research, stemming from this research, are presented.

Chapter 2

Control Theory

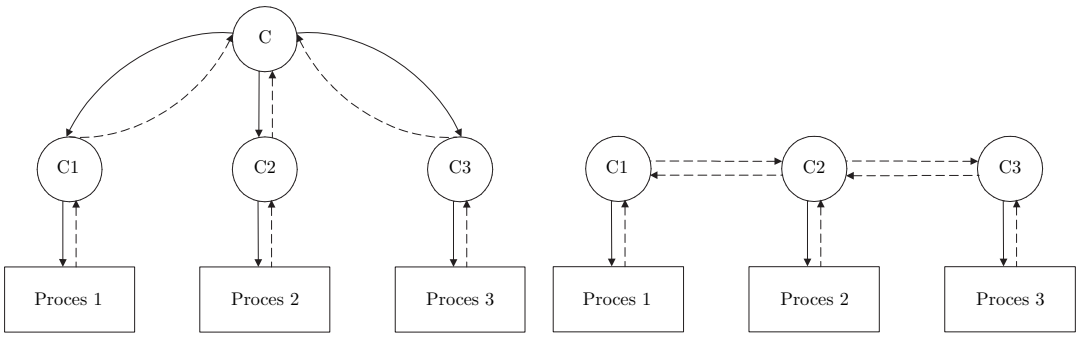
> *"All science is concerned with the relationship of cause and effect. Each scientific discovery increases man's ability to predict the consequences of his actions and thus his ability to control future events."*
> - Laurence J. Peter (1919 - 1988)

Since mankind always wanted to be able to control processes, this relationship between cause and effect has become a central theme in every research field. When one knows exactly how a process works by measuring it, one can learn how to operate it and in a further stage even to control it. These processes can be anything, manufacturing systems included. This chapter begins with a brief literature review on control theory in manufacturing systems in general, which will melt into a more specific description of the control theory which is investigated in this research.

2.1 Classification of Control Frameworks

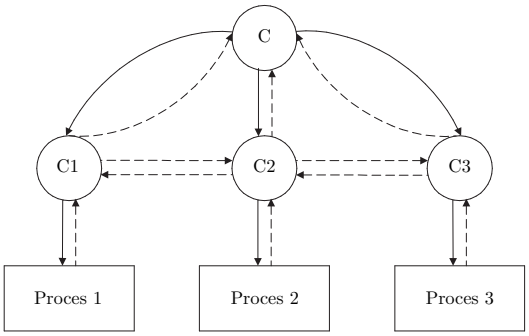
There are several classifications of control frameworks. Heragu *et al.* [Her02] provided a structural overview, which formed the outline for this section.
Duffie and Piper [Duf87] present a spectrum of architectures of a centralized controller, a hierarchical controller with dynamic scheduling, and a fully distributed heterarchical controller with intelligent parts. Lin and Solberg [Lin91] present four control paradigms: centralized information-centralized decision making, distributed information-centralized decision making, centralized information-distributed decision making, and distributed information-distributed decision making. Dilts *et al.* [Dil91] identify four basic control architecture forms for automated manufacturing systems: centralized, proper hierarchical, modified hierarchical, and heterarchical. They also summarize their characteristics, advantages, and disadvantages.

In this report, the control frameworks are classified into hierarchical, heterarchical, and hybrid control frameworks. The hierarchical frameworks map to the centralized and proper hierarchical classifications in [Dil91]. The hybrid frameworks include the modified hierarchical architecture in [Dil91] and their recent evolutions. All three frameworks are depicted in Figure 2.1, where the controllers are indicated with the letter C.



(a) Hierarchical framework.

(b) Heterarchical framework.



(c) Hybrid framework.

Figure 2.1: Three classifications of control frameworks.

The hierarchical framework assumes there is a hierarchy and a master/slave relationship between higher and lower levels of control. The hierarchy is introduced to handle the complexity of a manufacturing system. Sensory data flows in an upward direction from low level controllers (or unit controllers) at the lowest level to higher level supervisory controllers [Duf86]. Based on this, command data is generated and sent in a downward direction from supervisory controllers to units. Unfortunately, these sensory and command information flows have slow response times, which adversely affect the quality and timeliness of decisions in manufacturing since the environment at the time a control decision is executed is different from the one under which the decision was made. In addition, they almost completely ignore important interactions between unit controllers. Decisions are therefore made entirely by the master controller [Dil91].

The heterarchical framework focuses on interactions between unit controllers to allow system flexibility, while ignoring those between higher and lower-level controllers. The advantages include reduced complexity, high modularity, high flexibility and improved

fault tolerance. The contradiction problem between local objective and the overall system performance as well as deadlock detection and resolution problems are major drawbacks of this framework. Furthermore, a high degree of variability in the performance of a heterarchical system exits [Duf87, Duf86].

The hybrid framework has features of both hierarchical and heterarchical frameworks. Although hierarchical and heterarchical models have limitations, they also have several desirable characteristics. Some researchers have attempted to capture the positive aspects of both. These frameworks allow direct interactions among the low level controllers themselves as well as between high and low level controllers [Dil91].

This research includes further exploration of the work of Tolboom [Tol04] (see Section 1.1), and uses the hierarchical framework to control manufacturing systems.

2.2 Hierarchical Framework: Distinguishing the Layers

In the previous section, three types of control frameworks are discussed; a hierarchical, a heterarchical and a hybrid framework. The type of control strategy or method, which can be used in all the frameworks, has not been examined until now. All layers must be provided with suitable controllers. The number of layers differs per framework. In this research a two layer framework is chosen, since the expectation is that this situation already shows the working of a hierarchical system. Also, both layers can be physically interpreted. The choice of all controllers is important, only this research focusses merely on the high level controller.

High Level Control

High level control focusses on global, system wide optimization of the manufacturing system with a long horizon and low resolution. It translates predefined goals into meaningful assignments for underlying controller(s), based on the aggregated state of the system. Many approaches can be used for high level control, like the infinite capacity relation (MRP-I), the finite capacity relation (MRP-II), search heuristics to solve the optimization problem like ‘branch and bound’ and ‘beam search’ and mathematical programming to solve the optimization problem. Since this research starts with the same idea of control as Tolboom [Tol04] did, model predictive control is chosen as controller for the high layer. The characteristics of MPC are explained in Section 2.3.

Low Level Control

Low level control aims at local optimization of the manufacturing system, using a short horizon and high resolution. Research on low level control is clearly decomposed into four different classes by Fowler [Fow02]. These are sequencing rules and input control,

deterministic scheduling algorithms, control-theoretic approaches and search heuristics. As stated before, the emphasis of the control framework of this research lies on the high level control, and therefore a non-complex low level controller is chosen, namely flow rate control and sequencing policies. These are used to determine the optimal moment of lot release and the optimal sequence of lots. More information about these policies is provided in Section 2.4.

Two Layer Hierarchical Framework

In the previous two subsections, both control layers are filled. The combined framework results in hierarchical model predictive control. Previous research on hierarchical MPC has been performed by Sousa and Pereira [Sou94] and more recently picked up by Balduzzi [Bal01], Song *et al.* [Son02] and Vargas-Villamil *et al.* [Var03]. By combining the information which was previously presented and from these articles, a composite framework can be derived. This framework is presented in Figure 2.2.

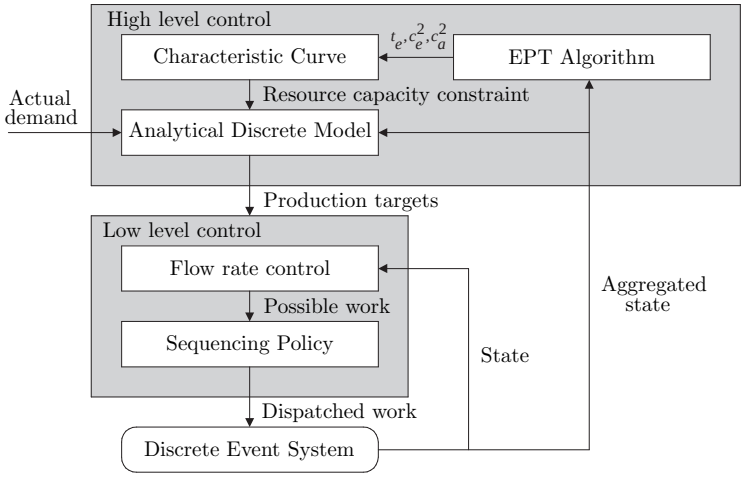


Figure 2.2: Two layer hierarchical model predictive control framework.

The above figure depicts the control framework which is used in this research. Note that this framework is more extensively worked out in Chapter 4. The general control framework for a manufacturing system was already presented in Figure 1.1c. Figure 2.3 shows how these two frameworks are related to each other. Both models contain a discrete event model or system. The right conversion block in Figure 1.1c is not explicitly present in the new framework. The controller is obviously represented by both control layers, but the high level controller performs most control work, whereas the low level controller forms merely the conversion from control to the discrete event model (left conversion block in Figure 1.1c). Finally, the continuous approximation model is also encapsulated in the high level control layer. It is made up by the EPT algorithm, the characteristic curve and the analytical discrete model.

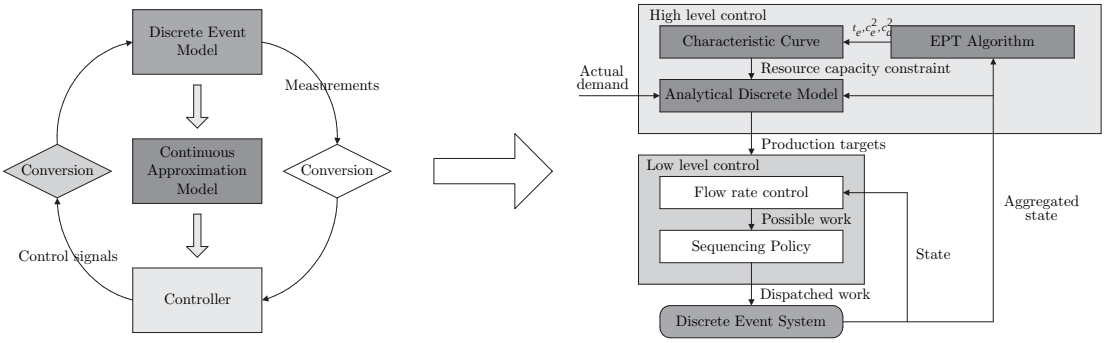


Figure 2.3: Similarity of the general dynamic control framework for manufacturing systems and the two layer hierarchical model predictive control framework.

2.3 Model Predictive Control

The high level control layer is provided with model predictive control, as already stated in Section 2.2. The principles of MPC are in detail explicated by Camacho and Bordons [Cam03]. Here, only a general overview is given.

MPC is a model based advanced control strategy, that has a significant and widespread impact on industrial process control. The term model predictive control does not designate a specific control theory, but a very ample range of control methods which make an explicit use of a dynamical model of the process to obtain the control signal by minimizing an objective function. MPC is 'predictive' because it uses this model to generate predictions of the future behavior of the process. Based on these predictions, an objective function is optimized with regard to the future inputs of the process. In this sense, MPC is an optimal controller with respect to the chosen objective function. Although prediction and optimized inputs are computed over a future horizon, only the new values of the inputs for the next sample are actually implemented and the same computational procedure is repeated during this next sample. This mechanism is known as a moving or receding horizon strategy.

The explicit use of a finite prediction horizon in the control problem is the most unique feature of MPC. The prediction horizon introduces feed forward control, that is, the MPC controller is able to take control action at the current time step, in response to a forecast of a future error between the reference and the actual output. This anticipation capacity is one of the advantages of MPC. The implementation of this control strategy is shown in Figure 2.4. Another advantage is the constraint handling capability, since constraints can be explicitly incorporated in the optimization. Furthermore, because of the straightforward optimization, MPC can deal with a high degree of interaction between inputs and outputs.

Obviously, there are also some drawbacks associated with MPC. The most important one is the need for an appropriate model of the process to be controlled. The performance

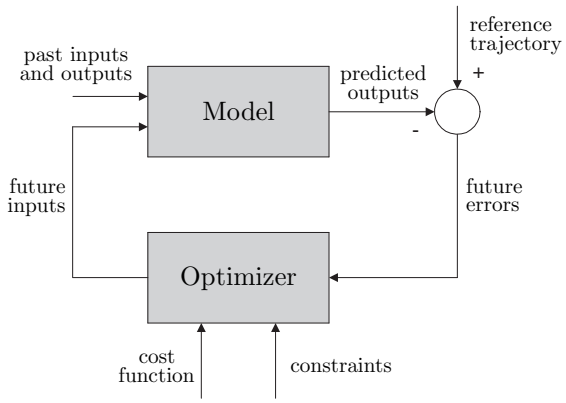


Figure 2.4: Basic structure of MPC.

of MPC strongly depends on the accuracy of the available model. Secondly, MPC is computationally rather demanding and, therefore, only applicable for relatively slow processes. Thirdly, the performance of MPC strongly depends on the values of tuning parameters including weighting factors, length of horizons, and sample intervals.

Because of its attractive capabilities MPC is widely introduced in the process industries. The disadvantages are not prevailing as time constants are usually large in industrial processes, so computation time is not an issue. Moreover, the profit of dynamic models has been recognized, not only for control but also for design and steady state optimization. Finally, small efficiency improvements can be very beneficial in industrial processes because of the high volume throughput.

2.4 Flow Rate Control and Sequencing Policies

The lower layer of the control framework contains a flow rate controller and sequencing policies, as already mentioned in Section 2.2. These approaches are used to determine the optimal sequence of lots and the optimal moment of lot release into the facility. Note that these two actions are not feedback control actions, but feedforward control.

The flow rate controller translates the production targets, which are delivered by the high level controller, into a set of lots (work) that are allowed to be processed. This set of lots is called possible work, since it has not yet been dispatched. Before allowing the set of lots to be actually sent, the order of the lots in this set can be rearranged by a sequencing policy.

A characteristic feature of sequencing policies is their myopic nature; sequencing rules review only the local state of the workstation, sometimes including pre-defined bottleneck stations. Since the choice on what to process next at a certain workstation significantly influences the performance of the downstream workstations, sequencing

rules are not likely to obtain a global optimum. Many sequencing rules exist, the most commonly used include First-In, First-Out (FIFO), Earliest Due Date (EDD), Shortest Process Time (SPT), Shortest Remaining Process Time (SRPT) and Least Slack (LS). Which one must be used in this research cannot be determined now, it is very dependent on the manufacturing system and the diversity in its products. After sequencing the set of lots, they can be dispatched to the system.

2.5 EPT Algorithm

As already mentioned in Section 2.2 and more explicitly shown in Figure 2.3, the conversion of the aggregated state of the system to the characteristic curve is performed using an effective process time algorithm.

It is stated by Gershwin [Ger89] that events that correspond to the discrete event system can be described by continuous variables. These variables can be treated as though they are deterministic. The approach is to define a set of variables for every activity. This set represents the behavior of the system in an aggregated way. It consists of three variables which represent the characteristics of the discrete event system. These variables are the mean effective process time t_e, the squared coefficient of variation of the effective process time c_e^2, which are both calculated by an EPT algorithm, and finally the squared coefficient of variation of the arrival time c_a^2. A much more comprehensive description of the concept of EPTs is given in Chapter 3.

The variables t_e and c_e^2, calculated by the EPT algorithm, and c_a^2 together determine the characteristic curve of the workstation under examination. This curve captures the highly non-linear relationship between throughput δ and work in process w, reflected by Figure 2.5. In queueing theory this relation is described by the approximation of Pollaczek-Khinchine [Pol30, Khi32, Tij94]. The characteristic curve is used as a non-linear constraint for the model predictive controller of the high level controller.

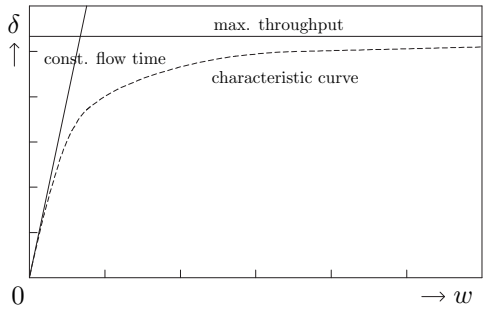


Figure 2.5: Example of a characteristic curve of a workstation.

2.6 Résumé

This chapter contains a brief literature review on different control techniques. First of all, three classifications of control frameworks are distinguished, these are hierarchical, heterarchical and hybrid. The hierarchical framework assumes there is a hierarchy and a master/slave relationship between higher and lower levels of control. The heterarchical framework focuses on interactions between unit controllers to allow system flexibility. The hybrid framework has features of both hierarchical and heterarchical frameworks.

This research uses a hierarchical control framework, consisting of two levels. High level control focusses on global, system wide optimization of the manufacturing system with a long horizon and low resolution whereas low level control aims at local optimization of the manufacturing system, using a short horizon and high resolution. High level control is performed by model predictive control (MPC) and low level control is carried out by flow rate control and sequencing policies. A far more detailed description of this control framework is provided in Chapter 4.

MPC generates predictions of the future behavior of the model, based on a optimization algorithm which is carried out for a future horizon, using the aggregated state of the system as input. This aggregated state is converted by an effective process time (EPT) algorithm into the characteristic curve of the system, which is used as a non-linear constraint for the high level optimization.

This conversion step, performed by the EPT algorithm is a very important aspect of the control, since the characteristic curve must very securely represent the actual system. Many EPT algorithms exist, but they are not all applicable to any system. That is why an extensive survey of effective process times is given in the next chapter.

Chapter 3

Effective Process Time: a Survey

"Time is the most valuable thing a man can spend."
- Theophrastus (372 BC - 287 BC)

The Greek philosopher Theophrastus noted this about 2300 years ago, and it still holds today. When time really is the most valuable thing you can spend, learning to make time work for you should be priceless. Only how can man let time work for him? The answer to this question lays in the definition of time. Especially in manufacturing systems, where time is used as a parameter for performance measurement. In Figure 2.2, the aggregated state of the discrete event system is used as input parameter for the high level controller. This aggregated state consists of raw data of measured times, which can be converted into the characteristic curve of the workstation (see Figure 2.5), which forms the input for the optimization algorithm of the high level control layer. This conversion is performed by an algorithm and when it is sensibly chosen, you can let time work for you.

The suggested algorithm is the effective process time algorithm. The introduction of EPT as a concept to account for throughput losses and process time irregularities on workstations has been made by Hopp and Spearman [Hop01]. Many members of the Systems Engineering Group have contributed to the field of EPTs. In this chapter, a survey will be given of the work they have performed.

3.1 Effective Process Time, the General Concept

The effective proces time of a lot refers to the total time seen by a lot on a machine from a logistical point of view [Hop01]. The idea of EPT is that it includes all sources of production capacity consumption on the workstation during the time that the workstation could have been processing the lot. EPT thus includes all time losses due to failure, setup, and any other source of variability. A similar description is given by

Sattler [Sat96], who defined the effective process time as all cycle time except waiting for another lot. It includes waiting for machine down time and operator availability and a variety of other activities. This concept has been visualized by Kock [Koc03] and is shown in Figure 3.1, where one machine cycle with all possible time losses is depicted. The concept of blocking, which is shown in the figure below, is defined and handled in Section 3.3.

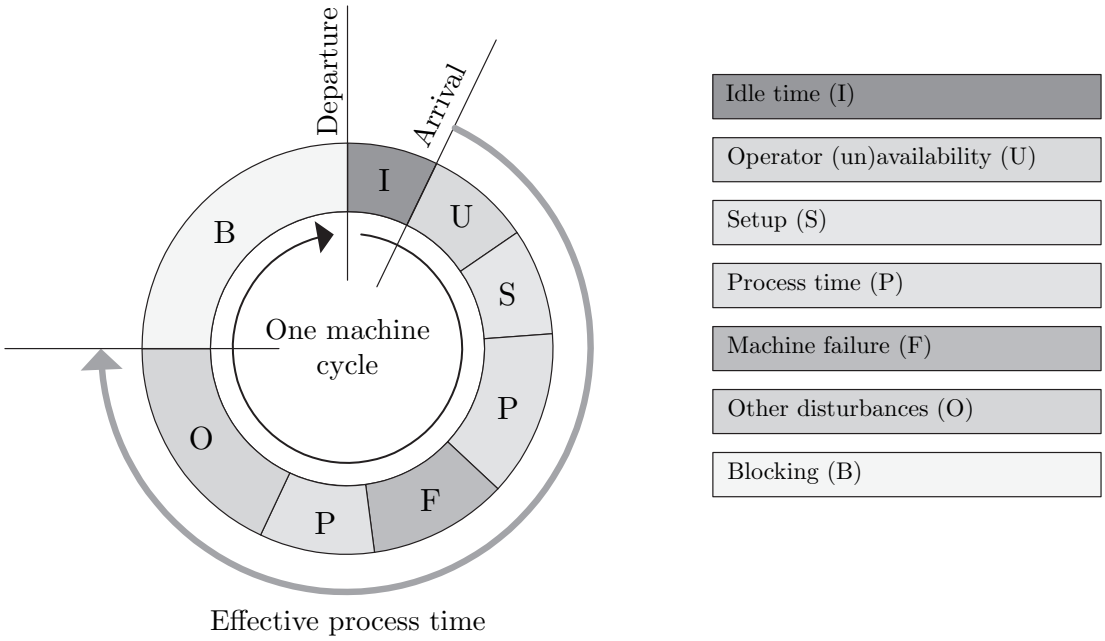


Figure 3.1: EPT Circle.

The main use of EPTs is as a performance measure, possibly in combination with overall equipment efficiency (OEE). But in this research, EPT calculation is mostly used for determination of the characteristic curve, which is used as a non-linear constraint for the optimization process in the high level controller.

Several EPT algorithms exist, all applicable for different situations, like finite or infinite buffered workstations, blocking or not blocking, single or multiple lot machines. In the subsequent sections, the main EPT algorithms encountered in literature are discussed. Note that only the algorithms, which are relevant for this research, are worked out. Others are only mentioned briefly.

3.2 Single and Multiple Lot Machines without Blocking

This section handles all EPT algorithms which can be used in case of workstations with infinite buffers. Also blocking is not considered in these algorithms.

1SLM

The most elementary EPT algorithm has been developed by Jacobs *et al.* [Jac03]. This algorithm is applicable to an isolated machine workstation which consists of an infinite buffer and One Single Lot Machine. Furthermore, some assumptions are made for proper usage. First of all, EPT realizations on a workstation are independent and identically distributed. Secondly, no preemption is applied when lots arrive. Finally, a machine is never idle if at least one lot is present at the workstation.

The event data used as input for the algorithm consists of actual arrivals $\mathbf{AA}_{i,j}$ (i.e. the moment that lot i physically arrives at workstation j) and actual departures $\mathbf{AD}_{i,j}$ (i.e. the moment that lot i physically departs from workstation j). An illustration of these events and the accompanying EPT realizations is provided in Figure 3.2.

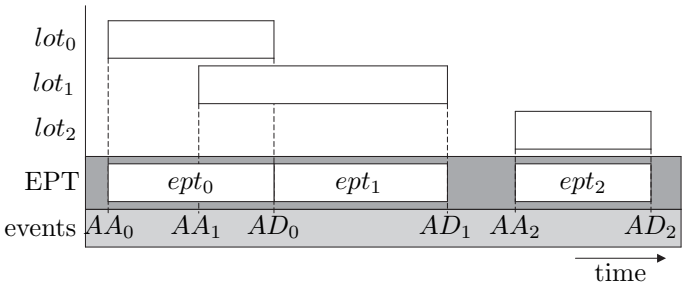


Figure 3.2: Gantt chart for 1SLM.

The EPT realization of lot i on workstation j is started at $\tau^s_{i,j} = \max(\mathbf{AA}_{i,j}, \mathbf{AD}_{i-1,j})$, depending whether the lot enters an empty or non-empty workstation. It is ended at $\tau^f_{i,j} = \mathbf{AD}_{i,j}$. Now, $\mathbf{EPT}_{i,j}$, being the EPT realization of lot i on workstation j, is captured by (3.1):

$$\mathbf{EPT}_{i,j} = \tau^f_{i,j} - \tau^s_{i,j} = \mathbf{AD}_{i,j} - \max(\mathbf{AA}_{i,j}, \mathbf{AD}_{i-1,j}). \tag{3.1}$$

Jacobs *et al.* [Jac03] translated this equation into the 1SLM algorithm, which is shown in Figure 3.3. Note that a bit knowledge of the formalism χ [Kle01] is needed to read this figure. Input parameters are the type of event ev and the corresponding occurrence time τ. The number of lots is denoted by n and s represents the start time of an EPT realization. After initialization, a repetitive loop is entered. If a lot arrives at an empty workstation ($ev = \mathbf{AA}$ and $n = 0$), an EPT realization is started. If a lot arrives at a non-empty workstation ($ev = \mathbf{AA}$ and $n > 0$), no action is taken. If a lot departs ($ev = \mathbf{AD}$), the EPT realization is recorded. If the workstation is empty after a departure, no action is taken, otherwise a new EPT realization is started.

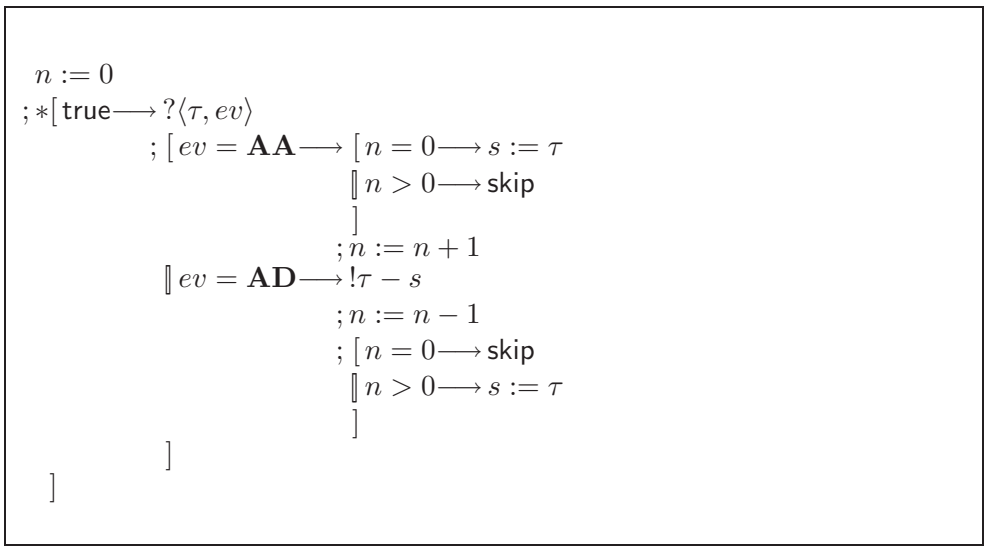


Figure 3.3: Algorithm 1SLM.

mSLM

Algorithm 1SLM can be extended for infinitely buffered workstations with m parallel, identical machines. EPT realizations are computed per machine, but are combined into a single EPT distribution since the machines are identical. The same assumptions as made for 1SLM, are applicable to this case. Additionally, lots are assumed to be dispatched in FIFO order. Provided that lots processed by machine j do not overtake one another, the EPT realization of lot i on machine j of workstation k is visualized in Figure 3.4 and represented by (3.2):

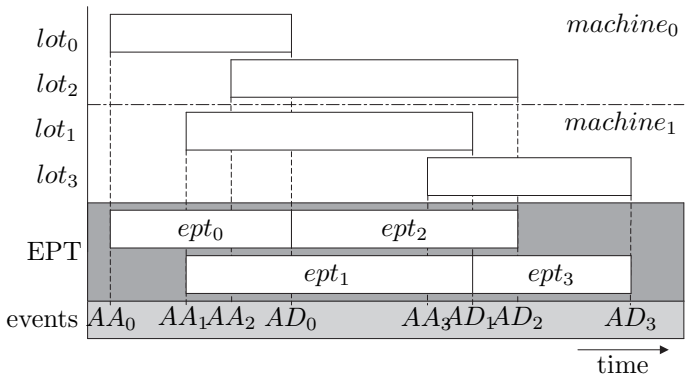


Figure 3.4: Gantt chart for mSLM.

$$\mathbf{EPT}_{i,j,k} = \mathbf{AD}_{i,j,k} - \max(\mathbf{AA}_{i,j,k}, \mathbf{AD}_{i-1,j,k}). \tag{3.2}$$

This equation is presented in algorithmic form by Jacobs *et al.* [Jac03] and is shown in Figure 3.5. This algorithm is self explanatory, when comparing it to the 1SLM algorithm of Figure 3.3, since the only new parameter is the machine j.

$$
\begin{array}{l}
nt := \langle 0 \rangle^m \\
;*[\,\mathsf{true} \longrightarrow ?\langle \tau, ev, j \rangle \\
\qquad ;\,[\,ev = \mathbf{AA} \longrightarrow [\,nt.j = 0 \longrightarrow st.j := \tau \\
\qquad\qquad\qquad\qquad [\!]\, nt.j > 0 \longrightarrow \mathsf{skip} \\
\qquad\qquad\qquad\qquad] \\
\qquad\qquad\qquad\quad ;nt.j := nt.j + 1 \\
\qquad\quad [\!]\, ev = \mathbf{AD} \longrightarrow !\tau - st.j \\
\qquad\qquad\qquad\quad ;nt.j := nt.j - 1 \\
\qquad\qquad\qquad\quad ;\,[\,nt.j = 0 \longrightarrow \mathsf{skip} \\
\qquad\qquad\qquad\qquad [\!]\, nt.j > 0 \longrightarrow st.j := \tau \\
\qquad\qquad\qquad\qquad] \\
\qquad\quad] \\
\quad]
\end{array}
$$

Figure 3.5: Algorithm mSLM.

mSLM - extensions

Within the Systems Engineering Group, multiple extensions have been developed for the mSLM algorithm. They are of no use for this research since they are developed for special occasions. For completeness of the survey, they are mentioned here, but only very briefly.

Rooney [Roo02] has contributed to the development of the mSLM algorithm by introducing the concept of m unequal parallel machines (instead of m identical machines) and exceptional first lots (i.e. requiring a setup). So now, EPTs are sorted per machine, distinguishing first lots from general lots. Another technical feature of this algorithm, called mSLM-Rooney, is the use of lists for a shorter notation. The algorithm will not be provided here.

An alternative extension of mSLM is presented by Jacobs *et al.* [Jac03], which allows violation of the EPT non-idling assumption. It is possible for a lot to wait for production on machine j whereas machine $i \neq j$ is still idle. Effectively, this means a loss of processing capacity since the lot could have been in process on machine i. This loss has to be included in EPT realizations. Unfortunately, this algorithm is not yet fully developed, since an analytical equation describing the length of an EPT realization, like the ones in the previous subsections, is not yet available for this situation.

The third extension of mSLM is to replace the m machines by batch machines, on which up to r different batch recipes can be processed. For this situation, EPT realizations can be computed using one of the algorithms presented above. Prior to application of an EPT algorithm, lot arrivals and lot departures must be translated into batch arrivals and batch departures. This transformation is done using algorithm mVBM-r (Multiple Batch Machines with Variable batchsizes and processing multiple Recipes) as presented by Van Bakel *et al.* [Bak03].

3.3 Single Lot Machines with Blocking

Unlike the already mentioned EPT algorithms, the algorithms discussed in this section can cope with blocking and finite capacity workstations.

1SLMB

All work before Kock [Koc03] stated that an EPT realization is defined as all time that a lot *claims* of a workstation. This includes blocking, since no new product can be processed, while the finished product is still *claiming* the production capacity. So the product is not actually *consuming* the production capacity, while it is blocked, only *claiming* capacity. Therefore, this blocking time must not be included in the EPT realization. Blocking can be a result of a control action, so it does not necessarily depend on the behavior of the workstation. So an EPT realization is now defined as all time that a lot *consumes* production capacity of the workstation. Would the realization persist while *claiming* production capacity, the lot is penalized for a capacity claim for which it is not to blame.

The moment that workstation j could have started processing lot i, i.e. that workstation $j-1$ finished processing the lot is called the possible arrival $\mathbf{PA}_{i,j}$. The moment that workstation j finishes processing lot i and tries to send the lot on is logically called possible departure $\mathbf{PD}_{i,j}$. Together with the actual arrivals $\mathbf{AA}_{i,j}$ and actual departures $\mathbf{AD}_{i,j}$, these four events are visualized in Figure 3.6. This figure represents two sequential, unbuffered workstations where workstation j blocks lot_1 on workstation $j-1$. The four used events in Figure 3.6 obey the following constraints:

$$\begin{aligned} &\mathbf{PA}_{i,j} = \mathbf{PD}_{i,j-1}, \\ &\mathbf{AA}_{i,j} = \mathbf{AD}_{i,j-1}, \\ &\mathbf{PA}_{i,j} \leq \mathbf{AA}_{i,j} \leq \mathbf{PD}_{i,j} \leq \mathbf{AD}_{i,j}. \end{aligned} \tag{3.3}$$

Consequently, an EPT realization begins at the moment that the workstation could have started processing the lot. The realization commences at the possible arrival if this occurred at an empty workstation ($\mathbf{PA}_{i,j} \geq \mathbf{AD}_{i-1,j}$), or at the actual departure

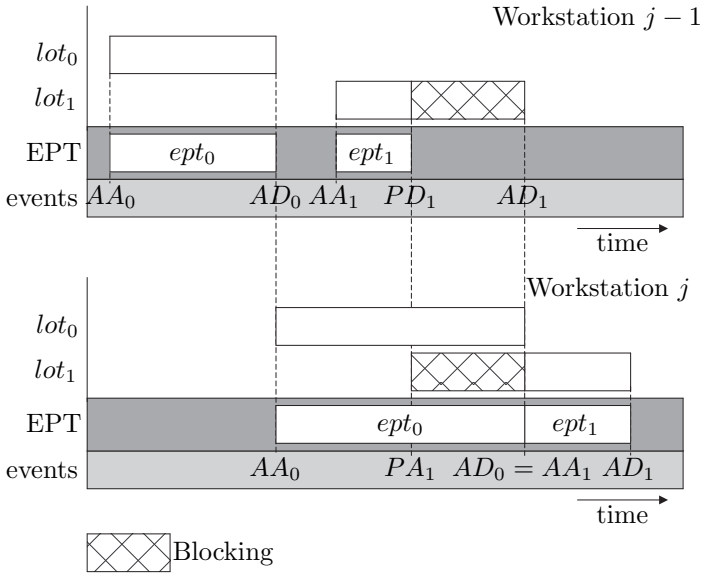


Figure 3.6: Gantt chart for two sequential, unbuffered workstations.

of the previous lot ($\mathbf{AD}_{i-1,j} \geq \mathbf{PA}_{i,j}$). The start time of the realization can thus be computed by $\tau^s_{i,j} = \max(\mathbf{PA}_{i,j}, \mathbf{AD}_{i-1,j})$. The EPT realization comes to an end at the moment the lot stops consuming production capacity of the workstation. Due to blocking, this may be well before the lot departs from the workstation, would the realization persist, the lot is penalized for a capacity claim for which it is not to blame. Consequently, the EPT realization ends at $\tau^f_{i,j} = \mathbf{PD}_{i,j}$. The EPT realization is then represented by (3.4):

$$\mathbf{EPT}_{i,j} = \tau^f_{i,j} - \tau^s_{i,j} = \mathbf{PD}_{i,j} - \max(\mathbf{PA}_{i,j}, \mathbf{AD}_{i-1,j}). \tag{3.4}$$

Wullems [Wul02] presented algorithm 1SLMB for the computation of the corresponding EPT realizations. The algorithm is applicable to unbuffered workstations with One Single Lot Machine, subject to Blocking. The algorithm assumes that events are presented in their order of occurrence, with event type $ev = \mathbf{A}$ (arrival) or $\mathbf{D}$ (departure) and the possible time of occurrence tmp and actual time of occurrence tma. The algorithm is presented in Figure 3.7 and equivalent to (3.4) if lots are not allowed to overtake one another.

The algorithm is very similar to 1SLM of Figure 3.3, besides the difference of tmp and tma and τ, the n is updated at different positions. The algorithm initializes and subsequently enters a repetitive loop. If a lot arrives at an empty workstation ($ev = \mathbf{A}$ and $n = 1$), an EPT realization is started at tmp where, in this case, $tmp = \mathbf{PA}_{i,j}$, whereas no action is taken if the workstation was not empty ($n > 1$). If a lot departs ($ev = \mathbf{D}$), the EPT realization is recorded, ending at $tmp = \mathbf{PD}_{i,j}$. If the workstation

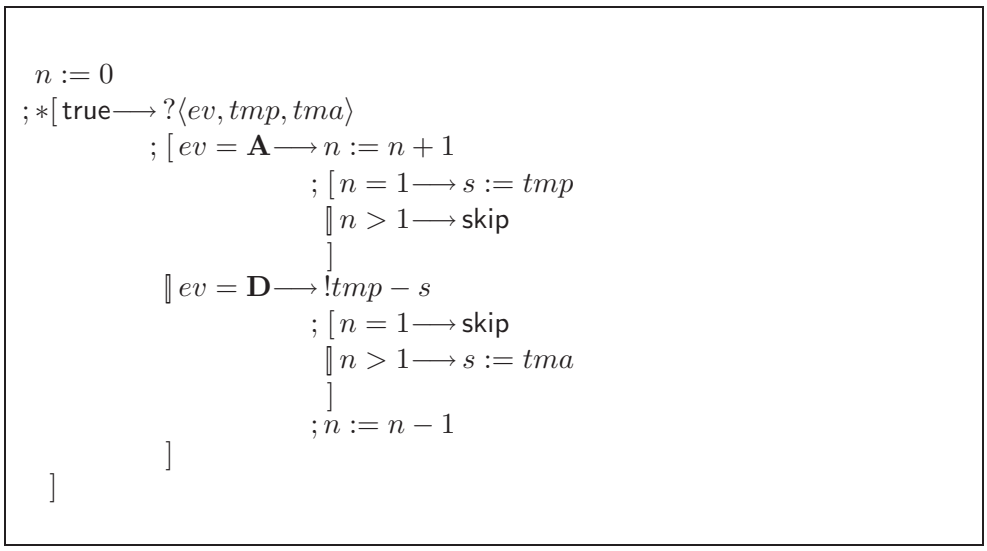


Figure 3.7: Algorithm 1SLMB.

remains empty, nothing happens. If the workstation is not empty, the realization of the next lot is started at the actual departure of this lot, i.e. $tma = \mathbf{AD}_{i,j} = \tau^s_{i+1,j}$.

The presented equation and algorithm have been derived for unbuffered workstations. When the underlying assumptions are expanded, one is able to apply 1SLMB to finitely buffered workstations. The extra assumption is that, after a workstation is done processing a lot, it is instantaneously sent on to the following workstation when this workstation has sufficient buffer capacity available. This is known as the assumption of instantaneous transport. For increasing buffer sizes, finitely buffered workstations will resemble infinitely buffered workstations. Larger buffers result in a reduction of the amount of blocking, until no blocking is left for infinitely buffered workstations. As stated before, blocking is the sole cause for a difference between a possible and an actual event. For infinitely buffered workstations, one thus knows that $\mathbf{PA}_{i,j} = \mathbf{AA}_{i,j}$ and $\mathbf{PD}_{i,j} = \mathbf{AD}_{i,j}$. Substituting this into (3.4) leads again to (3.1).

The algorithm 1SLMB was the first algorithm which considered blocking, but was still not sufficient for several blocking situations. Before looking at an improved algorithm, first the term 'blocking' is more clarified.

Blocking

Blocking has a negative impact on the throughput of manufacturing systems. It cannot be caught in one definition, since it is caused by several different aspects. Blocking can occur due to failure of (a part of) a manufacturing system, but also due to intentionally made choices of a designer or operator. According to Weber [Web03], blocking can be

subdivided in three categories: resource blocking, port blocking and condition blocking. Their descriptions are provided below.

Resource blocking Two tasks share one resource; they require the same resource for execution under the assumption that these tasks have the need for exclusive resource allocation. An example is the finite capacity of buffers, since the number of resources (buffer spaces) is not always big enough for the number of tasks to be performed (the number of lots that have to be stored).

Condition blocking A task has some condition on its input port, demanding this condition is fulfilled before activating the task. An example is when a controller intentionally leaves a machine idle, while it is empty and lots are waiting in the corresponding buffer.

Port blocking A port can be asynchronous or synchronous in nature. Asynchronous ports contain infinite buffers which do not lead to blocking. Synchronous ports, on the other hand, will propagate blocking throughout a transaction as the sender has to wait until the receiver is ready to communicate. Examples are the failure of an (unmodeled) transportation system, or a failure that occurred during idle time. Port blocking, by definition, violates the assumption of instantaneous transport.

1SLMB-us2d

Now the three types of blocking are known, they can be better dealt with in EPT algorithms. Kock [Koc03] has developed a new algorithm, based on 1SLMB, which is much better equipped to handle port blocking. Kock studied the results of the 1SLMB algorithm, when subject to port blocking. Three major observations have been made.

- From an EPT model point of view, port blocking is assigned to the downstream (receiving) workstation, while the upstream (sending) workstation causes this blocking.
- 1SLMB does not allocate port blocking if a lot has its possible arrival on a non-empty workstation. From a lumped parameter model point of view, all port blocking should be allocated, because it entails consumption of production capacity (since this is an unnecessary claim of production capacity).
- Port blocking realizations are combined with the effective processing part to form a single EPT distribution. A new distribution for port blocking should be derived, since port blocking differs from EPT in magnitude, frequency of occurrence and in general form of distribution.

The three errors of 1SLMB are corrected, by assigning port blocking to the upstream workstation, allocating all occurrences of port blocking and splitting the EPT realization

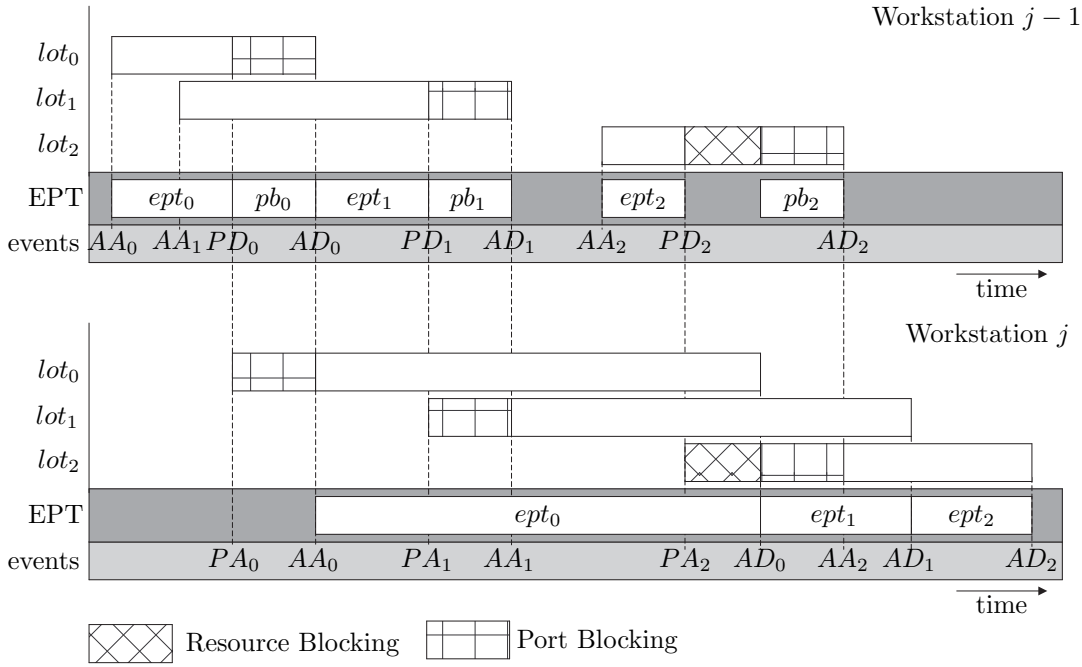


Figure 3.8: Gantt chart for 1SLMB-us2d.

in two parts, one for the service part ($\mathbf{EPT}_{i,j}$) and one for port blocking ($\mathbf{PB}_{i,j}$). The corresponding implications are reflected by Figure 3.8.

Following the previously derived equations, this situation is also caught in equation form. Now, two equations exist, one for each realization. The service part is represented by (3.5), which differs a bit from the 1SLMB algorithm (3.4), since the ($\mathbf{PA}_{i,j}$) as possible start time is replaced by ($\mathbf{AA}_{i,j}$). This is done to avoid that port blocking is included in the EPT realization.

$$\begin{aligned}\mathbf{EPT}_{i,j} &= \tau_{i,j}^{f_e} - \tau_{i,j}^{s_e} \\ &= \mathbf{PD}_{i,j} - max(\mathbf{AA}_{i,j}, \mathbf{AD}_{i-1,j})\end{aligned} \tag{3.5}$$

$$\begin{aligned}\mathbf{PB}_{i,j} &= \tau_{i,j}^{f_p} - \tau_{i,j}^{s_p} \\ &= \mathbf{AD}_{i,j} - max(\mathbf{PD}_{i,j}, \mathbf{BE}_{i,j})\end{aligned} \tag{3.6}$$

$$(\tau = \mathbf{AD}_{i,j} \wedge n_{i,j} = c_j - 1) \rightarrow \mathbf{BE}_{i,j} := \tau \tag{3.7}$$

The PB realization is captured by (3.6). This realization may commence when two conditions are fulfilled. First of all, the service part of the lot must have come to an end, thus $\tau_{i,j}^{s_p} \geq \mathbf{PD}_{i,j}$. Secondly, the buffer of the downstream workstation must not be saturated. Let $\mathbf{BE}_{i,j}$ be the time that the buffer of workstation $j+1$ last changed from

'saturated' to 'one buffer space available'. Then the second condition is represented by $\tau_{i,j}^{s_p} \geq \mathbf{BE}_{i,j}$. The PB realization ends at the actual departure of the lot, $\tau_{i,j}^{f_p} \geq \mathbf{AD}_{i,j}$.

A third equation is needed for more detailed specification of $\mathbf{BE}_{i,j}$. In (3.7), $n_{i,j}$ is the number of lots present at workstation j after arrival of lot i and c_j is the storage capacity of workstation j (i.e. the sum of the number of buffer spaces on the workstation and the single server). This equation implies that $\mathbf{BE}_{i,j}$ is only updated at an actual departure provided that the number of lots present on the workstation after the departure equals the total capacity minus one.

Equations (3.5), (3.6) and (3.7) are captured in the algorithm 1SLM-us2d, One Single Lot Machine, allocating port blocking to the UpStream workstation, providing Two Distributions. The algorithm is shown in Figure 3.9.

```
 nt := ⟨0⟩^m
;st := ⟨0.0⟩^m
;bet := ⟨0.0⟩^m
;pdt := ⟨0.0⟩^m
;*[ true ⟶ ?⟨ev, τ, j⟩
          ; [ ev = PA ⟶ skip
            [] ev = AA ⟶ [ nt.j = 0 ⟶ st.j = τ
                         [] nt.j > 0 ⟶ skip
                         ]
                       ; nt.j = nt.j + 1
            [] ev = PD ⟶ !⟨EPT, τ − st.j, j⟩
                       ; pdt.j = τ
            [] ev = AD ⟶ !⟨PB, τ − max(bet.j, pdt.j), j⟩
                       ; nt.j := nt.j − 1
                       ; [ nt.j = 0           ⟶ skip
                         [] 0 < nt.j < c.j − 1 ⟶ st.j := τ
                         [] nt.j = c.j − 1     ⟶ st.j := τ
                                                 ; [ j = 0 ⟶ skip
                                                   [] j > 0 ⟶ bet.(j − 1) := τ
                                                   ]
                         ]
            ]
 ]
```

Figure 3.9: Algorithm 1SLMB-us2d.

New variables are introduced; $c.j$ denotes the storage capacity of workstation j, $st.j$ the time that the newest EPT realization on workstation j started, $bet.j$ is the time that workstation i changed from having a saturated buffer to an unsaturated buffer, $nt.j$

denotes the number of jobs present at workstation j, $pdt.j$ the occurrence of the latest possible departure at workstation j and ev signifies the sort of event that has occurred. Finally, τ denotes the time of occurrence of the event under consideration.

During initialization, the values of nt, st, bet and pdt for all workstations j are set at 0. After initializing, the algorithm enters a repetitive loop. The first step in the loop is to read an EPT event. At a **PA**, no action is taken. If an **AA** occurred, an EPT realization is started if the lot entered an empty workstation, no action is taken if the workstation was not empty upon arrival. At a **PD**, the length of the EPT realization and the workstation where it occurred are printed. At an **AD**, the length of the port blocking realization and the relevant workstation number are printed. Note that for $j = 0$ an exceptional case exists, since WS_0 has no predecessor.

3.4 Validation of Algorithms

All algorithms discussed in this chapter have been validated analytically in their respective sources [Jac03, Roo02, Bak03, Wul02, Koc03]. Validation is performed by modeling an isolated workstation with exponentially distributed interarrival and process times. This model can be calculated mathematically by using queueing theory [Hop01] and experimentally using simulations. The mean throughput and mean flow time of the analytical and simulation model are compared for validation.

3.5 Résumé

EPTs are mainly used as a performance measure, possible in combination with overall equipment efficiency (OEE). But in this research, EPT calculation is mostly used as a constraint for the model predictive controller in the high level control layer.

Several EPT algorithms exist, all applicable for different situations, like finitely or infinitely buffered workstations, blocking or not blocking, single or multiple lot machines. The main EPT algorithms encountered in literature are discussed. These EPT algorithms are presented to guarantee a proper understanding of the subject matter, since in Chapter 5, the current form of EPT measurement is implemented into experimental cases.

But first the control framework, which is used for the experiments, is explained in the next chapter. The framework was already briefly explained in the previous chapter, but Chapter 4 gives a far more detailed view on the used hierarchical model predictive control framework.

Chapter 4

Control Framework

> *"The human brain must continue to frame the problems for the electronic machine to solve."*
> - David Sarnoff (1891 - 1971)

People like to frame things for a better understanding. When a difficult problem is put into a convenient arrangement or framework, it is more easily understood. Not only the understanding of a problem is important, also the solution of a problem is usually of great importance. This solving should be left to a computer and therefore the problem should be neatly framed. Sarnoff, a Russian-US inventor, already grasped this more than 30 years ago. For proper understanding and for computer-based solving, this research project also uses a strict framework. This control framework is already discussed in Chapter 2. It only has to be 'furnished' with the right tools.

The control framework which is used for this research project is a two layer hierarchical model predictive control framework. A clear visual representation of this framework is already given in Figure 2.2. In this chapter, the different blocks and communications in the mentioned figure are explained in detail. Some of these interpretations are copied from previous work, but some are introduced for this research. Also the relevance of the effective process times in this framework is clarified.

Before the control framework can be developed, first the system that has to be controlled must be specified. Because this research is merely about the improvement and implementation of the control framework as well as the adaption of the effective process time measurement, the system under control may be relatively simple.

4.1 Discrete Event System

The control framework is meant to control a manufacturing system, which is represented by a discrete event system. This system is in fact a model of a physical manufacturing

system and is represented as a χ-model (for more information about the specification language χ, see Kleijn [Kle01]). In this research, the manufacturing system at hand is not important, its only goal is to support the control framework and the applied control techniques. That is why a simple model has been chosen to work with. This model consists of a generator, two infinitely buffered workstations and an exit process (or stockroom). It is depicted in Figure 4.1. This is always the manufacturing system to which is referred during this thesis. The figure also denotes the throughput values x of each machine and the work in process levels w of each workstation. Note that x denotes the velocity of the products through the workstations, it is *not* an amount. These definitions are used throughout this thesis.

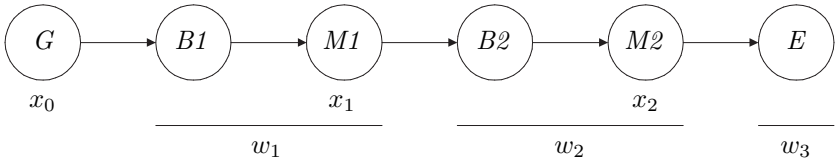


Figure 4.1: Simple line with indexes per workstation.

Many measurements can be carried out in this model, quantities like throughput, utilization levels, (effective) process times, buffer or wip levels, etcetera. Since the high level control layer is —contrary to the low level control layer— not directly implemented in the discrete event system, the number of needed measurements and communications to this layer is restricted. Therefore, it is important to find out which quantities are needed. Besides the outgoing data, the discrete event model (DEM) must also accept incoming control actions. Figure 2.2 already referred to the outgoing quantities as 'aggregated state' and the incoming quantities as 'production targets'. To see what information is needed, the next section describes the high level control layer in detail. The actual χ-model of the discrete event system is not be discussed until the next chapter (and its accompanying appendices), in which the experiments are handled.

4.2 High Level Control

As described in Section 2.2, the high level control layer is used as the primary step in the translation of actual demand into dispatched work. This part of the control framework is again depicted in Figure 4.2. Within this layer, planning approaches are used to derive a set of feasible production targets based on the actual demand and the aggregated state of the manufacturing system. The performance of the planning approaches depends heavily on the assumed relation for the resource capacity.

Tolboom [Tol04] approaches this resource capacity constraint by a set of linear constraints. For each planning cycle, the capacity constraint is determined in the working point using convergence techniques. The characteristic relation between work in process w and throughput δ is used by the planning approach and ensures the non-linear

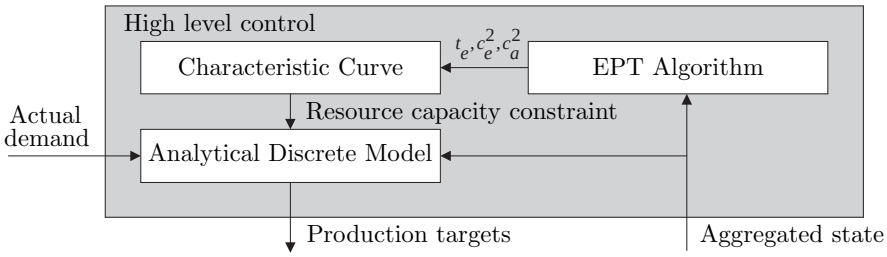


Figure 4.2: High level control layer.

degradation of system performance with increasing utilization. This set of resource capacity constraints is based on linear approximations of the clearing functions. In essence, these clearing functions (or characteristic curve) express the expected output as a fraction of the wip over a given period of time. Using queueing theory, this set is redefined into effective clearing functions. To illustrate these functions, Figure 4.3 is taken from [Tol04].

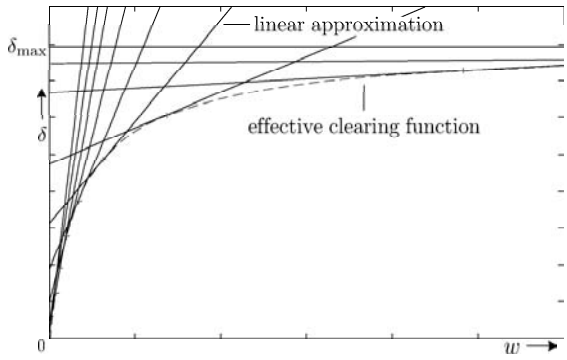


Figure 4.3: Set of linear approximations capturing the effective clearing function.

The effective clearing functions have several drawbacks, because they cause inaccuracies and the method is rather laborious. All clearing functions tend to slightly overestimate the resource capacity, this can also be seen in Figure 4.3, because when the number of linear approximations is limited, the gaps between approximations and actual curve becomes large.

To cancel out these shortcomings, this thesis introduces a new method for correct interpretation of the resource capacity. As said before, this can be captured by the characteristic curve, only this time, the direct non-linear curve is used. The next section handles this method.

4.3 Deriving the Characteristic Curve

The characteristic curve describes the unique relation between work in process w and throughput δ per workstation. Together with the cycle time φ, this relation is captured in Little's law:

$$w = \delta \cdot \varphi. \tag{4.1}$$

The above equation is further derived using queueing theory, which is introduced by Pollaczek-Khinchine [Pol30, Khi32, Tij94]. When a workstation is considered with m machines and completely general distributed interarrival times and process times, the following equation for the cycle time in the queue is obtained:

$$\varphi_q(G/G/m) = \left(\frac{c_a^2 + c_e^2}{2}\right) \cdot \left(\frac{u^\gamma}{m(1-u)}\right) \cdot t_e, \tag{4.2}$$

where $\gamma = \sqrt{2(m+1)} - 1$.

The notation $G/G/m$ is called the Kendall notation and describes the number of machines and the two mentioned distributions. In the future, this notation will be left away. The reader may assume that both times are generally distributed and that a workstation consists of a buffer and one machine, so $m = 1$. When the effective process time is added to the cycle time in the queue, the total cycle time of a lot at a workstation becomes:

$$\varphi = \varphi_q + t_e. \tag{4.3}$$

In the above equations, the utilization u is not yet explained. The u is captured by the fraction between the mean effective process time t_e and the mean interarrival time t_a. The interarrival time is in fact the reciproque of the throughput δ. In equation form:

$$u = \frac{t_e}{t_a} = t_e \cdot \delta. \tag{4.4}$$

When (4.1)–(4.4) are combined, the non-linearity of this characteristic curve becomes visible, because the throughput δ is squared:

$$\left(\left(\frac{c_a^2 + c_e^2}{2}\right) \cdot \left(\frac{t_e \cdot \delta}{1 - t_e \cdot \delta}\right) \cdot t_e + t_e\right) \cdot \delta - w = 0. \tag{4.5}$$

In order to completely define the characteristic curve, several parameters have to be known. The mean effective process times t_e and the corresponding squared coefficients of variation c_e^2 and interarrival times c_a^2 are essential parameters, see (4.5). How to correctly determine these parameters is explained in the next subsection.

Effective Process Times

Chapter 3 handles the EPT theory and containes several algorithms to compute the effective process times. These algorithms provide a list of EPTs, from which the t_e and c_e^2 can be calculated. For derivation of these two parameters, the mean and the variance of the distribution have to be calculated. Since only sample data is provided from the workstations in the model, the true mean μ and variance σ^2 are not known. However, the unbiased estimators of these quantities, m and s^2 respectively, can be computed using the sample data which contains n observed values:

$$m = \overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4.6}$$

and

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2. \tag{4.7}$$

To see whether the variability is small or large, the value of the variance relative to the mean value is of importance. Therefore, the squared coefficient of variation is defined:

$$c^2 = \frac{\sigma^2}{\mu^2}. \tag{4.8}$$

The next chapter deals with the exact implementation of the EPT measurements in the discrete event model. As discussed before, the old view on EPT measurement for the use as an input for this control framework could deliver wrong results. In the next chapter, the old view is presented and evaluated, as well as the new way of measuring EPTs for use as a control input parameter.

4.4 Deriving the Optimization Model

Now it is known how sample data is converted by the effective process time into the appropriate parameters to determine the characteristic curve, the last block of Figure 4.2 is examined; the discrete model or optimization model. This is the most important part of the high level controller, because this part performs the actual planning task.

This optimization model derives feasible production targets based on a prediction of the system dynamics by using model predictive control, as already explained in Section 2.3. In this research, the system dynamics —or more explicitly, the δ-w relation— are captured by the characteristic curve.

General Form

A discrete optimization model of a manufacturing system is a mathematical programming model that describes the state transitions with respect to a set of constraints. An important assumption which underlies such a model is the fact that time is discretized into fixed periods. In other words, the model describes the relation between the state x in subsequent periods. The general model of a constrained non-linear problem is defined as:

$$\begin{array}{lrcccl} \min\limits_{x} & f(x) & & & & \\ \text{s.t.} & x_L & \leq & x & \leq & x_U \\ & b_L & \leq & \mathbf{A}x & \leq & b_U \\ & c_L & \leq & c(x) & \leq & c_U \end{array} \tag{4.9}$$

where $x, x_L, x_U \in \mathbb{R}^n, f(x) \in \mathbb{R}, \mathbf{A} \in \mathbb{R}^{m_1 \times n}, b_L, b_U \in \mathbb{R}^{m_1}$ and $c_L, c(x), c_U \in \mathbb{R}^{m_2}$.

The first condition sets the lower and upper boundaries for the vector variable x, the second condition captures all linear equality and inequality constraints, whereas the third condition contains all non-linear equality and inequality constraints.

Definitions

Before the equations for the optimization tool can be derived, some definitions have to be fixed. This is necessary to gain more insight into the problem at hand. An important aspect of the model is that it uses information from two circumstances, a planned or expected situation and an actually realized situation. The expected situation cannot be measured, it automatically follows from the optimization process. The realized situation is measured at the end of a period and used as input for the optimization of the next planning horizon.

$x_k(t+i\|t)$	At time t planned target for workstation k for period $]t+i, t+i+1[$
$w_k(t+i\|t)$	At time t expectation of the wip of workstation k for time $t+i$, assuming that the planning is perfectly executed
$d(t+i\|t)$	At time t expectation of the demand for period $]t+i, t+i+1[$
$y(t+i\|t)$	At time t planned surplus in stock for time $t+i$, assuming that the planning is perfectly executed
$z(t+i\|t)$	At time t planned backorders for time $t+i$, assuming that the planning is perfectly executed
$x_k(t)$	Realized production at workstation k in period $]t, t+1[$
$w_k(t)$	Measured wip of workstation k at time $t^{(-)}$
$d(t)$	Realized demand in period $]t, t+1[$

The above definitions also present a clear view on the time scale which is used within the model predictive controller. Section 2.3 already mentioned that MPC uses the receding horizon strategy, so it optimizes an objective function over a future horizon. In this research, this horizon is divided into discrete periods. At time t, the optimization process is called to perform its task for the next p periods, and it uses the measurements of the system at the time $t^{(-)}$. This principle is explained in Figure 4.4, where two subsequent optimization runs are shown, each for five periods, so $p = 5$ and $i = 0, 1, 2, 3, 4$. The blue lines depict the first optimization cycle, which is performed at time t, the red lines represent the second optimization, performed at time $t + 1$. The figure shows that the new data of a new period can cause different optimal values of x for a period. Remember that only the first optimal value is actually implemented in the discrete event model, the remaining are used for feed forward control.

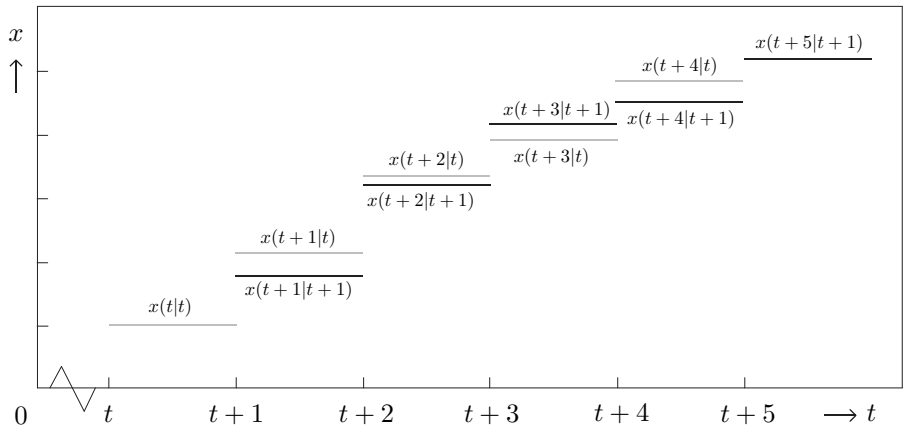


Figure 4.4: Example of a timeline for two optimization cycles.

Objective Function

Now that the system (Figure 4.1) and definitions are known, one can start with the actual derivation of the optimization model. First the objective function is derived. This function is presented as a cost function, which has to be minimized. The function is specially developed for this research, but has strong resemblance to other cost functions for manufacturing systems, as presented by Hackman and Leachman [Hac89] and Leachman [Lea01]. It schematically resembles all costs which are involved in the production of lots in a manufacturing system.

$$\min_{x(t+i|t)} \sum_{i=0}^{p-1} c_1^T \cdot \underline{\mathrm{x}}(t+i|t) + c_2^T \cdot \underline{\mathrm{w}}(t+i+1|t) + c_3 \cdot y(t+i+1|t) + c_4 \cdot z(t+i+1|t) \quad (4.10)$$

$$\text{with optimization variables } \underline{\mathrm{x}} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}, \ \underline{\mathrm{w}} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \ y \text{ and } z.$$

The cost function consists of four different kinds of cost, each with their own cost parameter. The first is the vector $\underline{c}_1$, which represents the production or release costs for all k workstations; $\underline{c}_2$ stands for the wip cost of the both workstations, so $k = 1, 2$. The stockroom can have two sorts of cost, c_3 denotes the stock cost when overproduction has taken place and c_4 denotes the back order cost when demand is not met. It is obvious that the choice for the weight of these cost parameters is an important one, since they have a great influence on the actual cost level. The actual figures which are implemented are provided later, for now it holds that $c_4 > c_3 > \text{mean}(\underline{c}_2) > \text{mean}(\underline{c}_1)$ for the obvious reason that backorders imply high costs, stockroom occupation a little less, but still more than the storage of semimanufactured products on the work floor. The release costs are the lowest of all.

Constraints

The optimization variables, which are used in the cost function, have to obey the following equations:

$$0 \leq x_0(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1 \tag{4.11a}$$

$$0 \leq x_k(t+i|t) \leq \text{char. curve} \qquad \forall t,\ i = 0, \ldots, p-1,\ k = \{1, 2\} \tag{4.11b}$$

$$0 \leq w_k(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1,\ k = \{1, 2\} \tag{4.11c}$$

$$0 \leq y(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1 \tag{4.11d}$$

$$0 \leq z(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1 \tag{4.11e}$$

Most of the above equations need no further explanation. The exception is (4.11b), whose upper boundary is determined by the characteristic curve. Recall (4.5), which contains the relation between work in process w and throughput δ. The non-linear constraint is now easy to derive. The optimization problem receives the measured wip levels w^* of each workstation. For each workstation, it applies that at this point w^* in the graph of the characteristic curve, a certain δ is defined. This is also the maximum throughput which can be reached in this situation. This is depicted in Figure 4.5, where the feasible choice for δ is indicated by the vertical arrow.

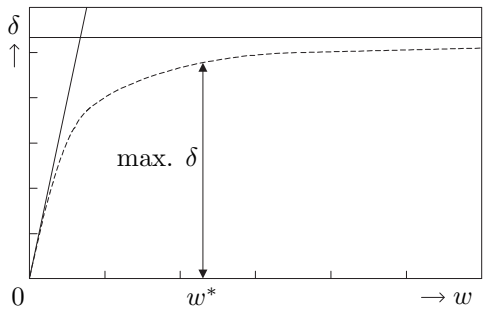


Figure 4.5: Feasible area of throughput δ for work in process level w^*.

When the above theory is applied to (4.1), it is obvious that the non-linear capacity constraint becomes:

$$w^* \geq \delta \cdot \varphi. \tag{4.12}$$

Using (4.5), the above equation is rewritten into:

$$\left((c_a^2 + c_e^2) \cdot t_e^2 \delta + 2t_e \cdot (1 - t_e\delta)\right) \cdot \delta - 2w^* \cdot (1 - t_e\delta) \leq 0. \tag{4.13}$$

Now that the capacity constraints are known, still some constraints are left to derive. These are equality constraints and are found by simply updating the wip levels of each workstation for every period. The last workstation, the exit process or stockroom, needs a deviant equation since no further production finds place, only the demand has to be reckoned with. In these equations, the new wip level is calculated by taking the old wip level, then adding the production of the previous workstation and subtracting the production (or demand) of the workstation itself. These relations may be seen as mass conservation laws and result in the following equations:

$$w_k(t+1) = w_k(t) + x_{k-1}(t) - x_k(t) \qquad \forall t,\ k = \{1, 2\}, \tag{4.14a}$$

$$w_3(t+1) = w_3(t) + x_2(t) - d(t) \qquad \forall t. \tag{4.14b}$$

The following relations also hold, they are trivial when looking at the definitions. Both $y(t+i|t)$ and $z(t+i|t)$ are positive numbers, when they are both zero, the wip level at the exit process is also zero.

$$w_k(t|t) = w_k(t) \qquad \forall t, k, \tag{4.15}$$

$$y(t+1|t) - z(t+1|t) = w_3(t) + x_2(t|t) - d(t|t) \qquad \forall t. \tag{4.16}$$

Note that in the optimal solution, y and z cannot both have a positive value, one of the quantities always has to be zero, resulting from the cost function.

When a deterministic system is considered, its behavior would be totally captured by (4.14)–(4.16). Since the manufacturing system under consideration is non-deterministic, the equations for the planning horizon can be derived using the three mentioned equations. At time t, $w_k(t)$ and $d(t)$ are known and the planning horizon i consists of p periods, so $i = 0, \ldots, p-1$. The actual equality constraints are found by rewriting (4.14):

$$w_k(t+i+1|t) = w_k(t+i|t) + x_{k-1}(t+i|t) - x_k(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1,\ k = \{1, 2\}. \tag{4.17}$$

By rewriting (4.16) the following constraints are found:

$$y(t+i+1|t) - z(t+i+1|t) = y(t+i|t) - z(t+i|t) + x_2(t+i|t) - d(t+i|t) \qquad \forall t,\ i = 1, \ldots, p-1. \tag{4.18}$$

Now, the total optimization model of this research is described. A comparison to the general optimization model, described by (4.9) with its three types of constraints is provided here. The first set of constraints capture the lower and upper boundaries of the optimization variables, in this research they are defined by (4.11) with the exception of (4.11b). The second set describes the linear constraints, which in this research are represented by (4.17) and (4.18). The third and last set captures the non-linear constraints, which are here described by (4.13).

A clear roundup of the optimization model as well as an extensively worked out example of an optimization run are provided in Appendix A.

4.5 Low Level Control

The low level control layer does not perform feedback control, but feedforward control, as already mentioned in Section 2.4. It still has an important role, because the production targets which are delivered by the high level controller are translated into possible work and eventually dispatched work. The low level layer of the control framework is again depicted in Figure 4.6.

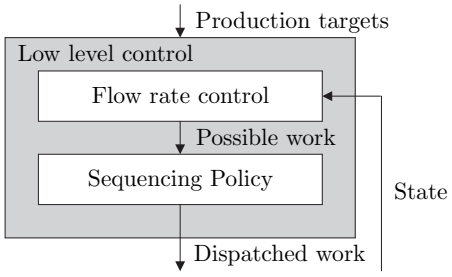


Figure 4.6: Low level control layer.

The translation of the production targets into possible work is done by a flow rate controller and forms a necessary step, which cannot be left out. This set of possible work can be rearranged by a sequencing policy into dispatched work, but this rearrangement is not necessary for good functioning of the controller. In fact, this is only needed when there is a difference between the lots that have to processed. In this research, all lots are intentionally equal, so no different process times are needed, no due dates are attached to the lots, nor any other differences exist.

Flow Rate Control

A flow rate controller translates production targets into dispatched work, which means that it actually determines the allowed departure times per workstation of the lots.

In this thesis, these are called the authorization times of the lots. They are equally distributed over a period, for a proper flow of the lots.

After each period and thus high level optimization, new production targets are received by the workstations. The amount of to-be-authorized lots is now known and the new period just starts, so the time is $t^{(+)}$. The first lot for this period may leave immediately, the others are divided equally over the period. This low level of control is encapsulated in the buffers of the manufacturing system. The working of the simple algorithm is presented as χ-code in Figure 4.7.

```
[ len(incs) > 0 ∧ n > 0 ⟶ temp := hd(incs)
                        ; incs := tl(incs)
                        ; [ n = rq ⟶ temp.3 := τ
                          [] n < rq ⟶ autht := lastauth + ta
                                    ; [ autht ≥ τ ⟶ temp.3 := autht
                                      [] autht < τ ⟶ temp.3 := τ
                                      ]
                          ]
                        ; lastauth := temp.3
                        ; n := n − 1
                        ; auths := auths ++ [temp]
[] len(incs) = 0 ∨ n = 0 ⟶ skip
]
```

Figure 4.7: Algorithm for the flow rate controller.

In the above χ-code, *incs* is a list which receives all incoming products. The variable n is initialized at rq, which is the optimal release quantity, received from the controller. If both mentioned quantities are positive, the following procedure is completed.
Because the sequencing rule at hand is FIFO, the head of *incs* is taken for authorization. When it concerns the first product of the period ($n = rq$), the authorization time, which is stored in *temp*.3 is chosen to be the present time τ, so the time when the period starts. When it does not concern the first product of the period ($n < rq$), the new authorization time *autht* is calculated by taking the last authorization time *lastauth* and adding the interarrival time *ta*. An extra selection is performed to actually store the authorization time in *temp*.3. This is needed, because sometimes the authorization time which is calculated could already have been passed, since the product could still be at the previous workstation. In such a case, the present time τ is stored in *temp*.3. After that, the *lastauth* is updated, as well as the variable n. Finally the list *auths*, which contains all products which have received an authorization time, is updated.

4.6 Résumé

The control framework globally consists of three parts; the discrete event system, the high level controller and the low level controller. In this thesis, the discrete event system is modeled as a simple manufacturing line, consisting of a generator, two infinitely buffered workstations and an exit process (or stockroom).

The high level control layer uses planning approaches to derive a set of feasible production targets based on the actual demand and the aggregated state of the system. It consists of a model predictive controller which optimizes a cost function, subject to a set of both non-linear and linear constraints.

The non-linear constraints are represented by the characteristic curves of each workstation, which are worked out in detail in this chapter. Important parameters of the characteristic curve are obtained from EPT calculations.

The linear constraints can be interpreted as mass conservation laws, since they capture the wip level behavior of the workstations. A very important aspect of these equations is that they describe the possible future behavior of the system, so each equation works with estimated variables which depend on the previous period.

Low level control consists of sequencing policies and flow rate control. Only the latter is implemented in the control framework in this research. The flow rate controllers are present in the buffers of the workstations and make sure the influx of the machines is each period equally distributed.

In the next chapter the presented framework is used for experiments to check whether the two goals of this research project can be achieved. The proposed hierarchical two layer model predictive control framework must be validated in an experimental environment. Secondly, the EPT measurements in casu condition blocking have to be reconsidered.

Chapter 5

Experiments

"In God we trust, all others have to prove it with facts."
- Mirko Nikolic

The above statement was picked up by the writer of this thesis while attending a masterclass at The Boston Consulting Group in Amsterdam. The Senior Vice President, Mr. Nikolic, spared some of his costly time to address the participants and this was one of his opinions on strategic consultancy. The presented statement is used to explain that customers not only desire a nice theory, but also need to see hard figures and facts to prove that the theory is well-founded. This is not only needed in a commercial surrounding, but also in the academic world proof is needed; even more than in the commercial world. This chapter deals with the experiments which are conducted in this research to show how the presented framework works in practice.

The previous chapter dealt with the 'furnishing' of the existing control framework, but the actual implementation of this framework in the specification language χ, the scripting language *Python* and the mathematical program *Matlab* is described in Appendix B. For this implementation, some extra assumptions and tuning parameters are needed, these are described in the next two sections. Subsequently, the results of the actual experiments are handled.

5.1 Simulation Assumptions

This section describes many assumptions which have been made in the several parts of the simulation framework. These assumptions concern the discrete event model and the high and low level controllers.

χ-model

The manufacturing system of Figure 4.1, consisting of a generator, two infinitely buffered workstations and an exit process, is modeled as a discrete event model in χ according to the schematic representation of Figure 5.1. Note that a more detailed version of this figure, with labeled communication lines can be found in Appendix B.3.

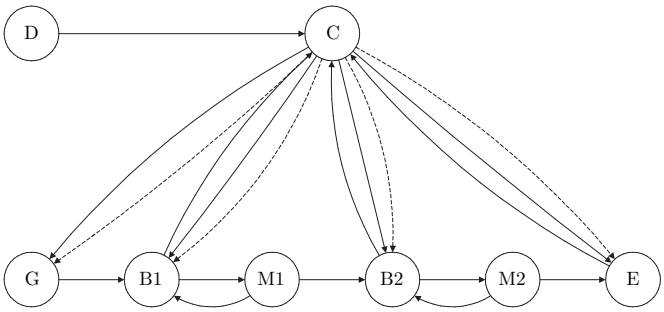


Figure 5.1: Simulation scheme in χ.

The above figure displays the demand generator D. This process is only a conduit, since the actual demand curve is generated by a *Matlab* script, which is explained in Appendix B.1. The controller C represents the high level controller. This process communicates with the *Tomlab* optimization scripts, which perform the actual control. A far more detailed description of the working of this discrete event model in χ is given in Appendix B.3. Some assumptions, which underly the model are provided here:

- The two workstations are both infinitely buffered (no resource blocking).
- Transportation of lots is instantaneous (no port blocking).
- Each workstation consists of one machine.
- Both workstations cannot break down.
- All lots are the same, no different process times, setup times, priority, etc exist.
- Production occurs in 24 hour shifts; 24 hours per day, 7 days per week.
- The production times per machine are generally distributed.

The two buffers in the χ-model contain the low level controllers. These controllers are developed under the following assumptions:

- No sequencing policy is present.
- Lot releases from buffer into machine are regulated. The first lot for a period may leave directly, the others are equally divided over the period. The times the lots may leave are called authorization times, as mentioned in Section 4.5.

- Lot releases from the generator are all at the same time, directly at the beginning of a period after the optimal release quantity has been determined.

Tomlab

The high level controller is modeled in *Tomlab*, which is an additional toolbox of *Matlab*. This controller contains the optimization algorithm and communicates with the process C of the χ-model through a *Python* script. The actual programming code of the m-files is provided in Appendix B.4. Here, only the most important assumptions are singled out:

- The allocation of costs in the objective function significantly influences the performance of the optimization problem. Therefore, cost should be wisely divided into release (or production) costs, wip costs, inventory and backlog costs.

- At the moment the production system is unable to meet the demand, a backlog is created. The change in backorders at the end of a period is defined as the difference between demand and the actual departures. A large penalty is assigned to backorders to ensure the optimization algorithm will avoid creating a backlog and will prioritize the elimination of an already existing backlog.

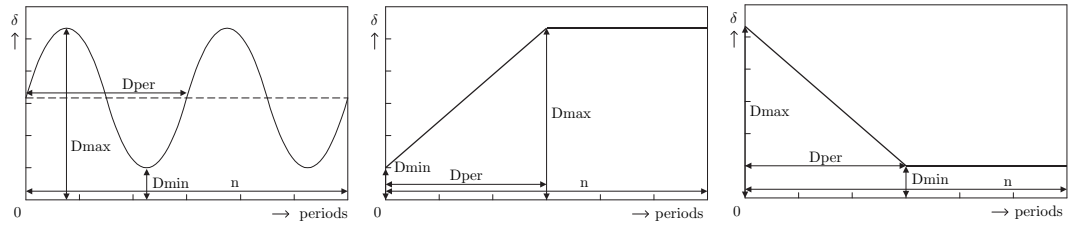
- A stockroom is placed at the end of the manufacturing line to meet overproduction, that is when the production is actually higher than the demand for that period. This can happen, since the optimization algorithm uses a receding horizon strategy and in this way can prevent a shortage in the future. Only this inventory in the stockroom also brings costs along.

5.2 Setup of Experiments

Tolboom [Tol04] experienced some problems with the control framework. This was due to the use of an incapable optimization toolbox and to condition blocking which corrupted the EPT measurements. The use of linear approximations of the non-linear characteristic curve had a minor contribution to the malfunctioning.
The performance of the newly 'furnished' framework is evaluated by conducting several simulation experiments. The first goal is to evaluate the general working of the control framework. A new feature which is implemented, is the non-linear characteristic curve as capacity constraint for each workstation. The second goal is to review the used EPT algorithm when subject to condition blocking. A new way of measuring EPTs has to be introduced. Experiments have to be performed to compare the performance of both EPT measurement implementations.

Varying the Demand

The only input parameters which can be altered are the demand, which varies over the time horizon and the process time distributions of both machines. The latter is rather difficult to vary and since for proper evaluation and comparison one of the input parameters should be kept constant, only the demand is varied. In this research, three types of demand are used as input for the simulation framework. The parameters of these three types are individually adjustable. The three types are a sine curve, a rampup and a rampdown trajectory. These demand trajectories with their adjustable parameters are shown in Figure 5.2. De parameters that can be adjusted in the *m*-file, are the total length of the horizon n, the type of demand *Dtype*, the minimum value *Dmin*, the maximum value *Dmax* and the (transition) period length *Dper*.



(a) Sine Curve. (b) Rampup. (c) Rampdown.

Figure 5.2: Three demand types.

Tuning Parameters

Besides the input parameters described in the previous subsection, there are also some tuning parameters, which can be altered to ensure a faster or more accurate calculation of the optimal solution each planning cycle.

An important factor in evaluating the MPC optimization problem, is the length of the planning horizon. A short horizon means that the estimation cannot function properly since the future demand is not enough reckoned with, but a long horizon can considerably delay the calculation time. An important aspect which has to be kept in mind when choosing the horizon length is to make sure the maximum lot demand over one period can be entirely completed during the planning horizon. This is needed, since the production of lots causes costs, and when the horizon is not long enough, the optimization process will not encounter the large cost factor of not meeting the demand that period; which means the optimization process will simply not produce at all, since in this way, the optimization tool believes no costs occur.
Information about the manufacturing system is needed for this; the mean production time for the first workstation is 0.21 hours and for the second workstation the mean production time is 0.23 hours. One planning cycle consists of 24 hours and the demand for one period can become 105 at the most. From these figures, it can be deducted

that the length of the planning horizon should at the very least consist of two periods ($105 \cdot (0.21 + 0.23) = 46.2$ hours ≈ 2 periods). But when a future fluctuation in demand exists, it is sensible to extend this length. In order to avoid problems, an ample horizon length of five periods is chosen; this is the standard length throughout this thesis.

Another parameter is the begin condition vector $\underline{x}_0$, which contains the starting values for the optimization and is free to fill in. With the standard horizon length of five periods, this vector consists of 35 elements. Endless possibilities exist to declare this vector, but simple reasoning leads to three main possibilities. The first is the standard *Tomlab* option, which is to use all zeros. The second option is to fill in the wip-levels, which are measured, at the appropriate places. The last is to fill in the demands for the coming periods of the planning horizon length at the appropriate places. Several test runs have been performed. They indicate that the last option works best for the simulations at hand, considering the calculation time of the optimization; the optimal solution was in almost all cases the same for each option. With these demands filled in at the right places, the starting vector itself is generally not feasible, but the optimization tool can handle this. Sometimes, no feasible solution exists because of high demands, together with low present wip-levels and low capacity (due to high measured EPTs). Fortunately, the optimization tool of *Tomlab* does not crash nor ends up in a lifelock situation, but limits itself to a maximum of 500 iterations. It then returns the values of the iteration where the toolbox ended, this may not be feasible, but the χ-model will receive practical production targets. This is also explained later during the experiments and in Appendix C.

A special choice which has to be made, is the choice of the appropriate *Tomlab* solver. *Tomlab* is an extensive toolbox with a lot of solvers which contain the optimization algorithms. Due to the nature of the problem at hand, that is a non-linear constrained optimization problem, the selected solver is the so-called *conSolve* solver.

The final property which influences the quality and speed of the optimization is the addition of extra constraints. The mass conservation laws are linear equality constraints, which are easy to solve for the *Tomlab* toolbox. The capacity constraints (or characteristic curves) however, are non-linear inequality constraints, which are a lot more difficult to solve. Therefore some additional linear constraints are introduced, which support the capacity constraints. Per workstation, two extra constraints are added, which represent the maximum throughput and the slope of the characteristic curve in the origin. These lines are shown in Figure 2.5. In fact, these two constraints are the two extreme linear approximations which Tolboom [Tol04] used for his problem, see Figure 4.3.

Performance Measures

Before one can start with the conduction of the experiments, the performance measures must be chosen first. This is needed to make sure the right data is saved in an output file for later evaluation. Important quantities to measure during simulation are:

- The buffer level per workstation.
- The production of the system with respect to the demand (the number of cumulative backorders).
- The effective proces times per workstation.

These measures are used to evaluate the control framework and to compare the performances using different ways of EPT measurement.

5.3 Testing the Control Framework

In this section, the answer to the first research objective is provided, which means that the improvement of the two layer model predictive control framework is evaluated here for its proper functioning. The performance of the framework is evaluated for the presented manufacturing system using different types of demand input. All tuning parameters are set up as described in the previous section.

Two remarks about the simulation runs must be made here. First, the experiments are conducted with the same process time distributions to ensure a fair comparison. Secondly, several simulation runs with different seeds are conducted to ensure reproducibility of the results. The seed of a distribution denotes the starting value of the samples which are taken from the distribution. For each seed value, the order of samples from a distribution is the same. The random number generator of χ is a linear congruential generator. If this generator is full-period, any choice of the initial seed will produce the entire cycle in some order. If, however, a generator has less than full-period, the cycle length could in fact depend on the particular value of the seed chosen. For now, it is important to realize that it is wise to chose high prime numbers as seed for the random number generator, which is proved by Hull and Dobell [Hul62].

Experiment I: Sine Curve, Low Utilization

First, the framework is tested using a sine curve as demand generator. The parameters used for the demand are as follows; the simulation length consists of 8500 periods, which is a little more than 200000 hours. This duration has been empirically determined during the test period, using the information of the conducted long test runs of the simulations. The minimum and maximum values of the sine are 40 and 80, this means that the maximum utilization does not exceed $u = \frac{80 \cdot 0.23}{24} = 0.77$, so no problems should occur here, despite the variability at hand. The period length is chosen to be 50. The outcome of the experiment is shown in Figure 5.3.

Figure 5.3 shows the fluctuations of the wip-levels of both workstations and the wip-level of the exit process. The latter has backlog when it is negative and a surplus when

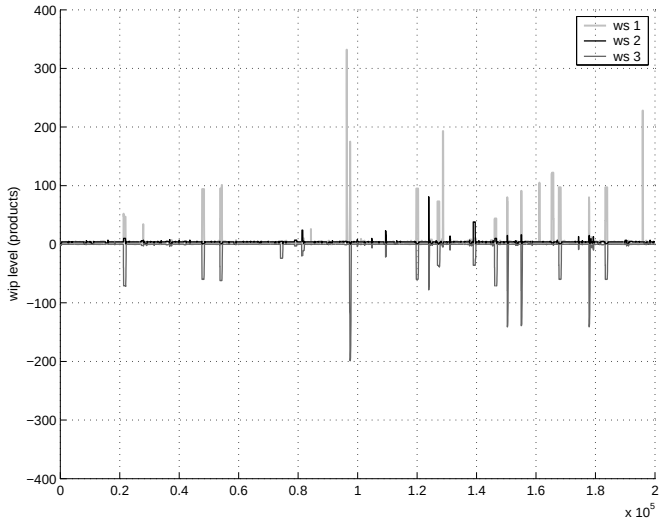
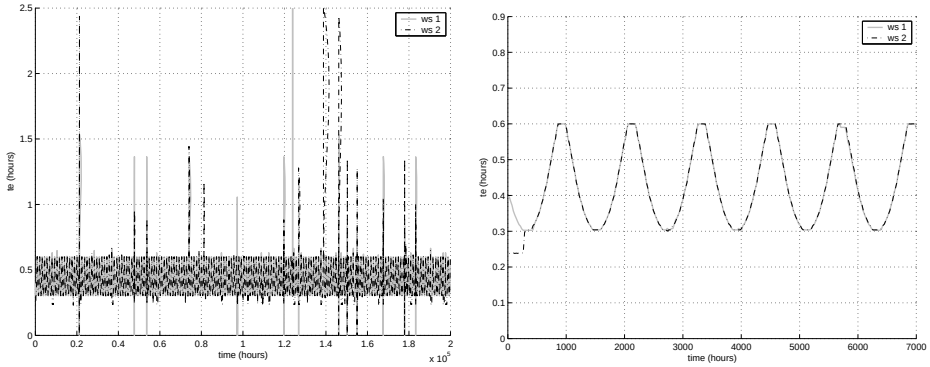


Figure 5.3: Experiment I: Wip-levels.

it is positive. From this figure, one can carefully conclude that the framework in general works. That is, the wip-levels of the buffers relatively do not rise very high, because the highest number is about 320, which is four times the maximum demand. Also the cumulative backorders are limited to 200, which means it takes at most three periods to clear this backlog. Sometimes, the wip-level of workstation 1 becomes suddenly very high, this is easy to explain. The optimization tool is not restricted at the generator, because this process is no machine with a maximum capacity, so sometimes the target for the generator can become high. Furthermore, all lots released by the generator are moved in total to the buffer of workstation 1 at the beginning of a period, this is possible because the buffers are chosen to be infinite. That is why sometimes the buffer (or wip) level at workstation 1 becomes that high.

Another interesting parameter to view in this experiment is the effective process time. Several times throughout this thesis, it is mentioned the old framework did not function properly due to incorrect EPT measurement in case of condition blocking. Therefore, the measured EPTs at both workstation are depicted in Figure 5.4a, with a zoomed area in Figure 5.4b.

Figure 5.4 clearly shows that the measured mean effective process times t_e of both workstations form a sine curve, exactly in counter phase with the demand. The effective process time of workstation 2 does not immediately convert in a sine curve, but after a short time it joins the curve of workstation 1. The t_e of workstation 1 should be 0.21 hours and the t_e of workstation 2 should be 0.23 hours. The wrong values of both t_e's in Figure 5.4 can however be declared. The demand fluctuates between 40 and 80 products and one period consists of 24 hours. Therefore, the t_e's fluctuate between $\frac{24}{40} = 0.60$ hours and $\frac{24}{80} = 0.30$ hours.



(a) Complete time horizon. (b) Zoomed area.

Figure 5.4: Experiment I: Measured EPTs.

The presented figure distinctly proves the EPT measurement is performed wrongly, since the effective process time is per definition supposed to be completely independent on capacity utilization. This erroneous EPT measurement is dealt with in Section 5.4, which handles the improvement of EPT measurement in casu condition blocking. For now, in spite of this measurement error, the control framework performs well for this case. But as already mentioned, the utilization level is chosen to be low. In Experiment IV, the same experiment is conducted, only the mean demand will be higher. Before this experiment is done, first another demand type is used for evaluation of the control framework.

Experiment II: Rampup, Low Utilization

This second experiment uses a rampup as demand, for a further evaluation of the control framework. This time, the total time horizon is many times smaller than in the first experiment, since once the rampup is completed, the demand is constant and further evaluation is not really needed. The minimum value is again 40, the maximum value is 80 and the transition length is 50 periods (1200 hours). Figure 5.5 shows the results.

Figure 5.5 shows the same quantities as Figure 5.3, only with completely different length of both positive as negative y-axes. Again, it turns out that at first sight, the control framework works as desired. The cumulative backorders are limited, as well as the wip-levels of both workstations. Curious is the fact that the backlog stays at four, since the machine should have enough capacity. Apparently, the optimization tool does not mind the fact that a constant backlog of four products exists.

The sine curve showed that the measured effective process times were incorrect, since they displayed a dependency on the capacity utilization. For this rampup experiment, the values are again shown, in Figure 5.6.

The results show again a utilization dependency of the effective process times. Especially

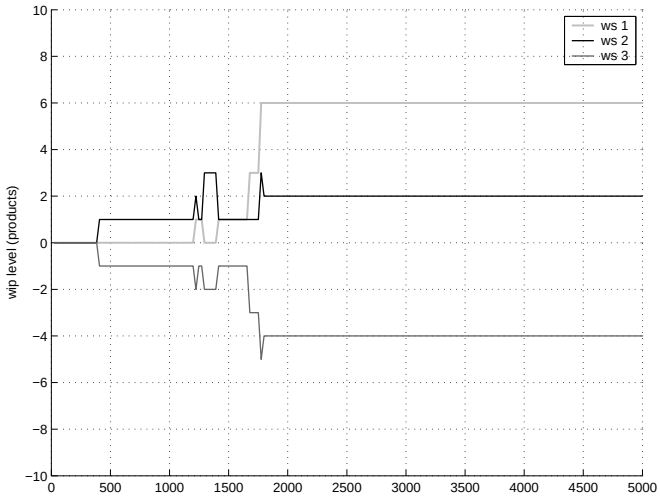


Figure 5.5: Experiment II: Wip-levels.

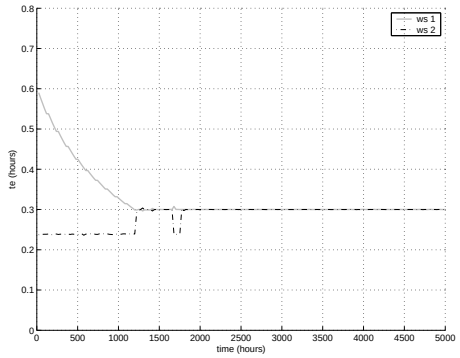


Figure 5.6: Experiment II: Measured EPTs.

the EPT of workstation 1 is not constant. It starts with a value which is two times as high as the eventual steady state value. The reason for this is explained by the fact that at first, less products are released into the line than in the steady state phase. So the utilization in the beginning is twice as less as at the end. This utilization dependency of the EPTs is dealt with in the next section, as previously mentioned.

Experiment III: Rampdown, Low Utilization

This third experiment uses a rampdown demand function. All parameters are chosen to be exactly the same as in Experiment II, the rampup situation. The results are shown in Figure 5.7.

Figure 5.7 shows that the wip-levels vary more frequent and stronger than in the second

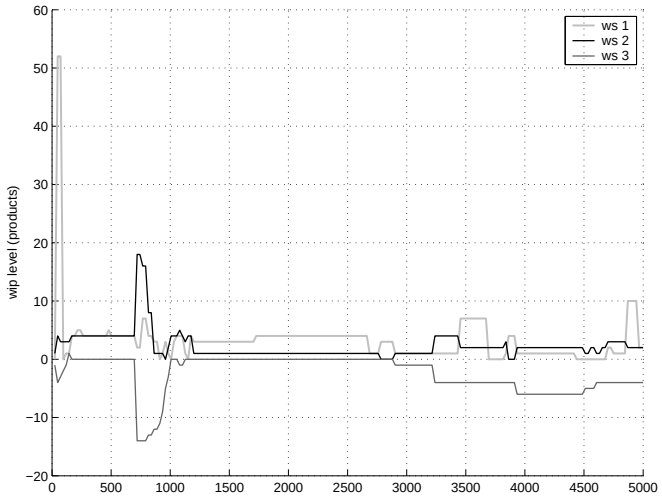


Figure 5.7: Experiment III: Wip-levels.

experiment. However in general, it shows no big surprises, the framework seems to work properly. To check whether the utilization dependency is also present here at the EPT measurement, Figure 5.8 is presented.

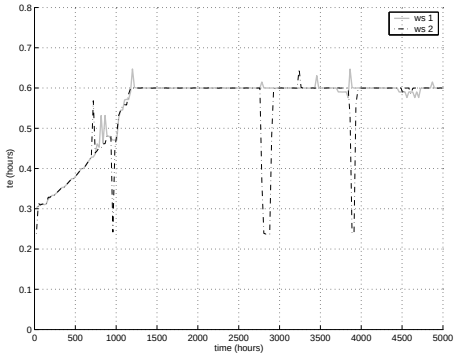


Figure 5.8: Experiment III: Measured EPTs.

Again, the results show a utilization dependency of the effective process times. This time both workstations show a utilization dependent behavior. After the rampdown is completed, they more-or-less stabilize.

Experiment IV: Sine Curve, High Utilization

For the previous experiments, the maximum utilization levels were intentionally chosen to be far below $u = 1$. With this experiment, a sine curve is used with a higher maximum utilization. This is done to determine whether the existing utilization dependency of

the EPTs has a negative impact on the performance of the control framework.

Again, a sine curve is used, because this demand type showed the dependency best. All parameters are the same as in Experiment I, only this time, the minimum and maximum values are 80 and 100, respectively. At the first experiment, the mean was 60, here it is 90. The maximum utilization has also grown from $u = 0.77$ to $u = 0.95$. The results of this experiment are depicted in Figure 5.9.

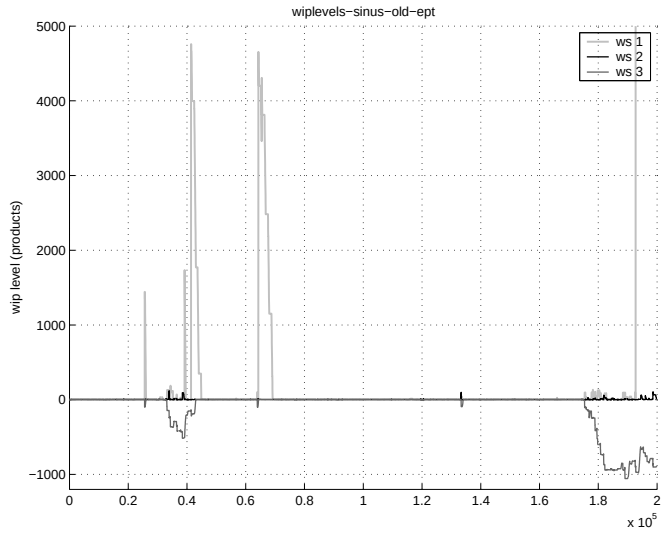
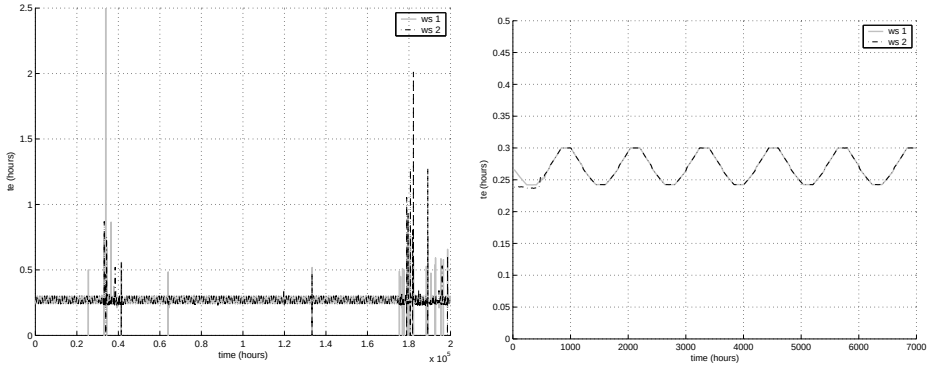


Figure 5.9: Experiment IV: Wip-levels.

When looking at the units at the axes of Figure 5.9, it is immediately obvious that the control framework performs much worse than in the low utilized experiment. The number of cumulative backorders is much higher than before and the wip-level of workstation 1 now sometimes contains several thousands of products, which means that the optimization tool has calculated these release quantities for the generator. For the explanation why this can occur, the output of the optimization tool of *Tomlab* is needed. This output can be written away in a so-called diary file, which is a standard feature of *Matlab*. A sample output can be found in Appendix C. When one looks at this output, it turns out that the optimization tool cannot find a feasible or optimal solution. In such a case, *Tomlab* returns the values, which it was evaluating at the time the optimization ended. These values can become extraordinary large, as can be seen in Figure 5.9.

The expectation is that the measured EPTs fluctuate just as much as before. The values are shown in Figure 5.10.

Again, the measured EPTs fluctuate in counter phase with the sine curve of the demand, as can be seen in Figure 5.10. The absolute values of the EPTs are lower than the ones presented in Experiment I, this is due to the fact that the utilization and throughput are higher, in combination with the erroneously dependency of the EPTs with these quantities.



(a) Complete time horizon. (b) Zoomed area.

Figure 5.10: Experiment IV: Measured EPTs.

Experiment V: Sine Curve, Extreme Low Utilization

Experiments I–III considered a manufacturing system with a low utilization level, this fifth experiment considers an even lower utilization. The goal of this fifth experiment is to show the vicious circle which occurs due to the wrong EPT measurements, as mentioned in Section 1.1. This means that the EPTs are measured too high because of the wrong EPT algorithm, which causes a too low estimation of the system's capacity. Therefore, the controller will set lower targets while it could set high targets. This means that the demand is not always met, which results in periodical backlog.

To show this phenomenon, a sine curve is used with the same parameters as before, only this time the minimum and maximum values are 10 and 50, respectively. The mean utilization is now $u_{mean} = \frac{30 \cdot 0.23}{24} = 0.29$, whereas the maximum utilization is $u_{max} = \frac{50 \cdot 0.23}{24} = 0.48$. No capacity problems should occur here, but when simulating, it becomes clear the control framework is pulled along in the vicious circle for almost each planning cycle. The results of this experiment are depicted in Figure 5.11.

Figure 5.11 only shows a small time interval, but this is only done to clearly show the behavior of the system. The y-axis depicts the demand and the difference between release quantity and demand for the last workstation. This last difference shows the problem of the too low estimated capacity of the system, in particular the last workstation. When the demand becomes 50, the calculated optimal release quantity should be the same, since the wip-level is also tuned to this purpose by the optimization tool. But when one looks at Figure 5.11, the shortage after each demand peak of 50 draws the attention. Each period, some periods heavier than others, the optimization process reacts not quickly enough to the demand fluctuation. This is depicted by the negative dark line in the figure. When the demand slowly decreases, the controller makes up the shortage by producing a surplus.

The above problem occurs when the capacity is estimated too low. The described vicious circle applies to the above case, the only reason it does not become worse is the

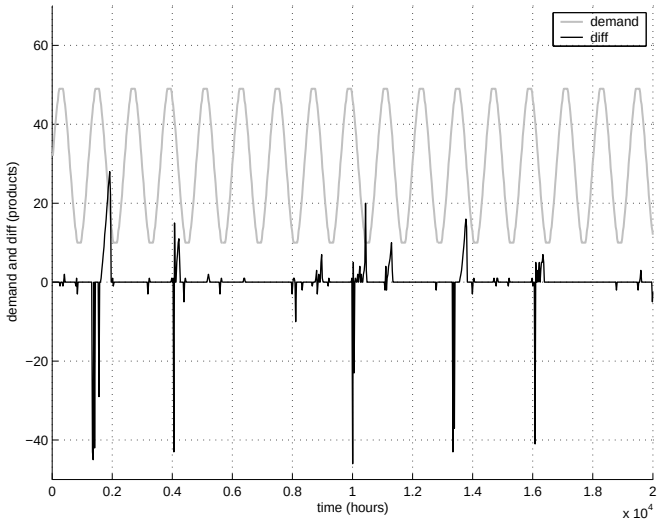


Figure 5.11: Experiment V: Demand and difference between release quantity and demand of the last workstation.

fluctuation of the demand to an ever lower utilization each time, so the controller has time to make up for his previously created shortage.

This section shows that the control framework in general works, it only uses a wrong measurement algorithm for the effective process times, since the system is subject to condition blocking. This causes a capacity utilization dependency of the EPTs, which is wrong in definition. Especially in a highly utilized system, this is disturbing and the control framework performs considerably worse is such cases. Also the presence of the previously discussed vicious circle is demonstrated. The next section describes a new proposal for EPT measurement which eliminates the described utilization dependency.

5.4 Improvement of the EPT Measurements

This section handles the improvement of the EPT measurements, which is the second research objection that is presented in Chapter 1. This means, the capacity utilization dependency is canceled out. Before the improved algorithm is presented, first the old statements about EPT measurement are repeated here. Then, the new algorithm is explained. New experiments are conducted to prove the working of this EPT algorithm.

Developing the new EPT Algorithm

The old algorithm, is the following one, which is already discussed in Chapter 3:

$$\mathbf{EPT}_{i,j} = \tau^f_{i,j} - \tau^s_{i,j} = \mathbf{AD}_{i,j} - \max(\mathbf{AA}_{i,j}, \mathbf{AD}_{i-1,j}). \tag{5.1}$$

The problem with the above equation is, that the arrival time at the buffer of the first workstation is equal for each product per period. Besides that, the EPT is dependent on the utilization, since the flow rate controllers spread the possible departure times over the period. When the machine is lowly utilized, the EPT becomes higher and vice versa.

To eliminate the above problem, the start time $\tau^s_{i,j}$ must be defined differently. In case of the current algorithm, the effective process time includes the waiting time of a product for his authorization time to become true. It is not fair to include this waiting time, because it is intentionally introduced by the combination of high and low level controller and thus may be seen as condition blocking, see also Section 3.3.

Therefore, a new approach is used. The authorization times, which are assigned to the products by the flow rate controller, are now used for the calculation of the EPTs. Note that the calculation of authorization times is already discussed in Section 4.5. The flow rate controllers are encapsulated in the buffers of the workstations. Such a buffer divides all products (or lots) over three lists.

- The first list is *incprods* (*incoming products*). All products which are received from an upstream workstation are immediately placed in this list. When necessary, the lots can be rearranged by a sequencing policy.
- When a new period commences, the high level controller determines the optimal release quantities and delivers them to the buffers. The flow rate controller then calculates the authorization times and then as many products as the calculated release quantity are put into the list *authprods* (*authorized products*), together with their authorization times.
- The list with authorization times is constantly evaluated and when a authorization time of a product passes, the product is moved to the third list, *outbprods* or *outbound products*. All products in this last list can leave the buffer immediately if the machine is not occupied and asks for a product.

The above described system with three lists makes it possible to introduce the new EPT algorithm. The times which accompany the product through these lists make it able to introduce a new quantity, the $\mathbf{AT}_{i,j}$, which stands for the authorization time of lot i at workstation j. These times are assigned by the flow rate controller in the buffer, as described. Then, the new equation, which is adapted from (5.1) would like this:

$$\mathbf{EPT}_{i,j} = \tau^f_{i,j} - \tau^s_{i,j} = \mathbf{AD}_{i,j} - \max(\mathbf{AT}_{i,j}, \mathbf{AD}_{i-1,j}). \tag{5.2}$$

It may seem like (5.2) has almost not changed from (5.1), but one must realize there exists an important difference. Besides the difference in the equation, it is important

to design the low level controller in such a way, that it explicitly gives each product an authorization time, which is stored in the product tuple.

Now the new EPT algorithm is deducted, it is time to prove its proper functioning. This proof is delivered within the three next subsections, which describe new experiments and their results.

Experiment VI: Sine Curve, Low Utilization, New EPT Algorithm

The first experiment which is conducted with the new EPT algorithm is a repetition of the very first experiment, a sine curve with low utilization. The exact same parameters for the demand curve are used as in Experiment I. To guarantee a fair comparison between both EPT measurement methods, the exact same seeds as in the first experiment are used. The presented figures under Experiment I and here, are the results of the simulations with the same seed. This experiment is conducted to make sure the control framework also works in this case. The results of wip-levels and measured EPTs are presented in Figure 5.12 and Figure 5.13, respectively. Note that the scaling of all x- and y-axes is different to those of Figure 5.3 and Figure 5.4. This is deliberately done to make sure the values and shapes of the curves are visible.

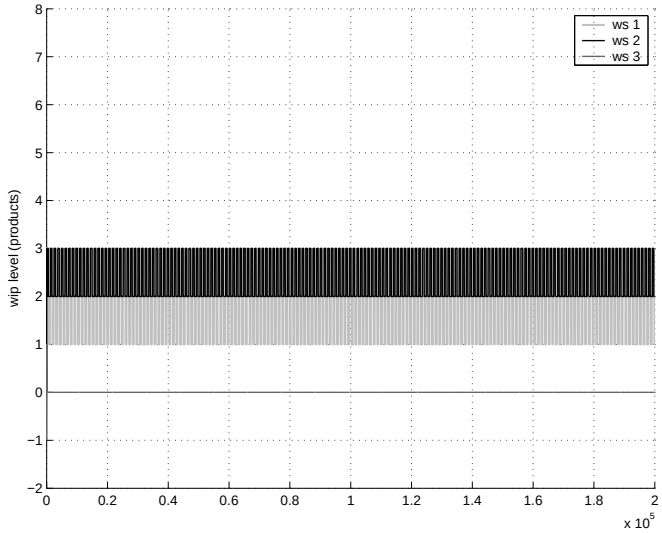
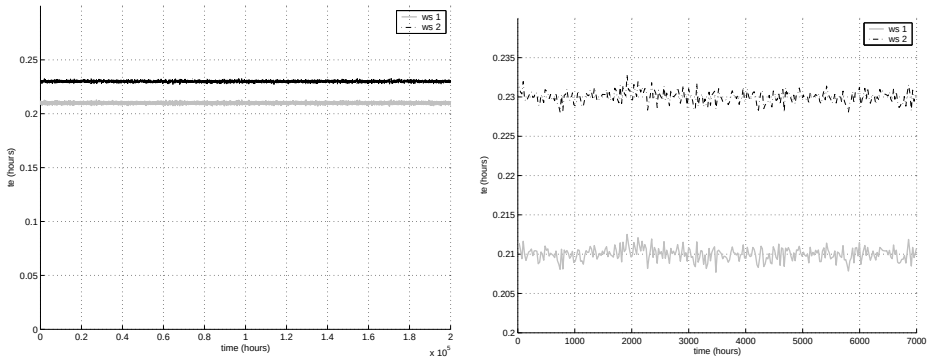


Figure 5.12: Experiment VI: Wip-levels.

From the presented figures it can be concluded that the control framework with the adapted EPT measurement works perfectly. The wip-levels are much lower than before and the cumulative backorders are, after initialization, constantly zero. Furthermore the EPTs turn out to be independent on the capacity utilization, like they are supposed to be. They vary a bit per period, but this is logical, since variability is at hand. One can clearly see that the mean process time of workstation 1 is 0.21 hours and of



(a) Complete time horizon. (b) Zoomed area.

Figure 5.13: Experiment VI: Measured EPTs.

workstation 2, it is 0.23 hours. This was not the case at all previous experiments when using the old EPT algorithm.

Now the control framework functions properly, higher utilization levels are introduced to check whether the framework again performs better than the last time in Experiment IV.

Experiment VII: Sine Curve, High Utilization, New EPT Algorithm

This seventh experiment is again a repetition of a previously conducted experiment, this time Experiment IV. Just like the last experiment, the same conditions, referring to the seed and demand type, hold. The results are shown in Figure 5.14 and Figure 5.15

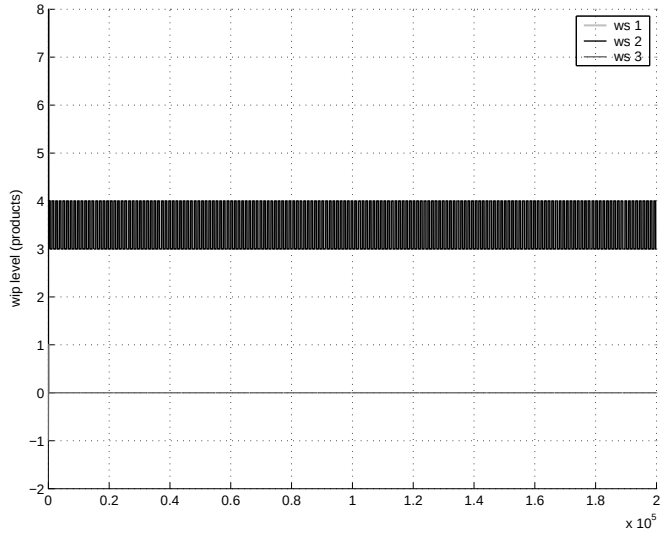
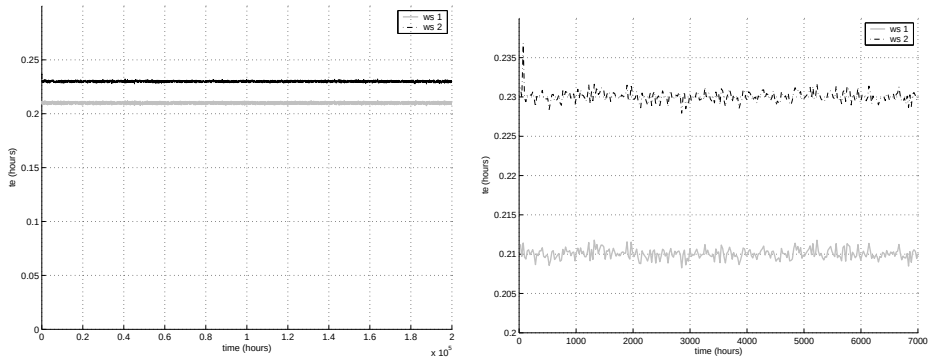


Figure 5.14: Experiment VIIa: Wip-levels.



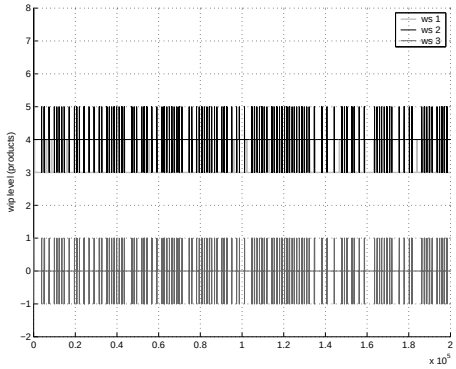
(a) Complete time horizon. (b) Zoomed area.

Figure 5.15: Experiment VIIa: Measured EPTs.

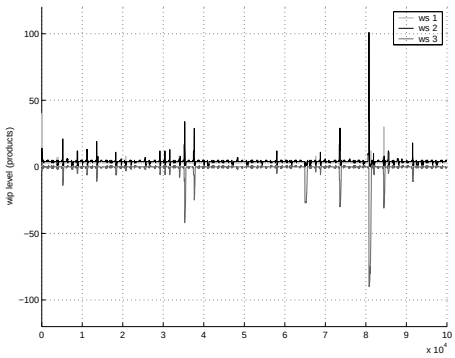
As one can see, the results have much improved compared to Experiment IV. The wip-levels are very low and the cumulative backlog becomes zero after a very short initialization period. The effective process times again show an independency of the utilization.

Now that it is empirically proved that the presented control framework with the new EPT algorithm works with the presented utilizations, it is interesting to look at even higher utilizations. Therefore, two extra experiments are conducted, VIIb and VIIc. Experiment VIIb uses a sine curve demand which is limited between 92 and 102, and thus has a mean utilization $u_{mean} = \frac{97 \cdot 0.23}{24} = 0.93$ and a maximum utilization $u_{max} = \frac{102 \cdot 0.23}{24} = 0.98$. Experiment VIIc uses a sine curve demand which is limited between 95 and 105, and thus has a mean utilization $u_{mean} = \frac{100 \cdot 0.23}{24} = 0.96$ and a maximum utilization $u_{max} = \frac{105 \cdot 0.23}{24} = 1.01$. Both experiments will not be commonly used in a real life setup, but these simulations are used to point out the capability of the framework. Only, when simulating VIIc, a calculation error occurred at the optimization level, which caused the simulation to stop before the time horizon of 8500 periods was reached. The results of Experiments VIIb & VIIc are shown in Figure 5.16. This time, only the wip-levels are depicted.

Figure 5.16a shows that Experiment VIIb still functions very well, although the utilization level has become very high. It even performs better than the low utilization of Experiment I, because the wip and backlog levels remain much lower than in the first experiment. But Experiment VIIc encountered some difficulties in the optimization tool. Many times, the problem became infeasible, just the same as explained in Experiment IV. The problem of infeasibility also occurred at other highly utilized experiments, but now it caused too many problems for the simulation to properly end. Therefore, only half the time horizon is presented in Figure 5.16b.



(a) VIIb: Sine Curve Demand between 92 and 102



(b) VIIc: Sine Curve Demand between 95 and 105

Figure 5.16: Experiments VIIb & VIIc: Wip-levels.

Experiment VIII: Sine Curve, Extreme Low Utilization, New EPT Algorithm

This eighth and last experiment repeats Experiment V, which had extremely low utilization levels. Only this time, the new EPT algorithm is used. The same diagram which was made for Experiment V is made for this simulation and is presented in Figure 5.17.

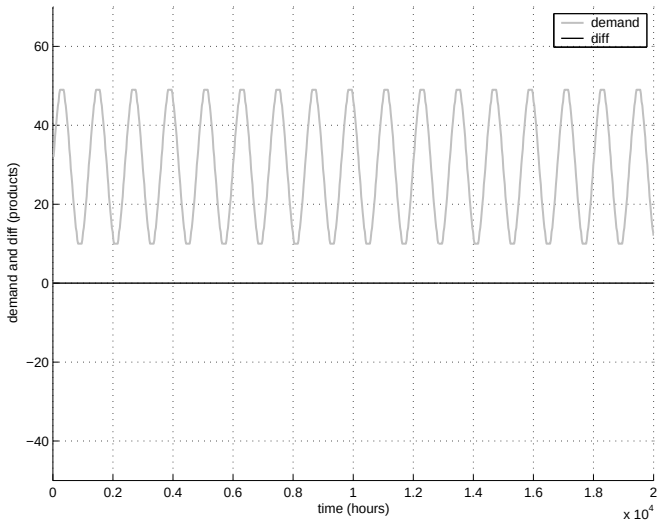


Figure 5.17: Experiment VIII: Demand and difference between release quantity and demand of the last workstation.

One of the aims of this research was to improve the EPT measurement, as described in this section. Not only highly utilized manufacturing systems benefit from this improvement, but also manufacturing systems with an extreme low utilization are better controlled. This is possible, since the discussed vicious circle is neutralized. Figure 5.17

shows a perfectly controlled system (the difference between releases and demand is constantly zero), even more when one compares it to Figure 5.11, with the results of Experiment V.

This section introduced a new way of measuring EPTs for proper use as a capacity constraint of the two layer hierarchical model predictive control framework. Due to this improvement of control, much better results are achieved with several types of demand and levels of utilization. The new method especially proved to be effective in a highly utilized setup.

5.5 Résumé

This chapter evaluates the control framework, which was derived in Chapter 4, with the use of a simple manufacturing line with two infinitely buffered machines. It focusses on two main subjects, where the first is to validate the control framework and the second is to improve the EPT measurement in case of condition blocking.

First, several assumptions and important choices are made for the χ-model of the manufacturing system, the implementation of the low level controller and the general setup of the high level controller, which is implemented in the *Matlab* optimization toolbox *Tomlab*.

A setup is determined how to conduct the experiments. There are several input and tuning parameters. The input parameter is declared as the demand type for the manufacturing system, which can be altered throughout the experiments. The tuning parameters are needed for a proper functioning of the optimization toolbox *Tomlab* and are determined by several preliminary test runs. Before the actual experiments commence, the performance measures are chosen. These are the buffer (or wip) levels of the workstations, the number of cumulative backorders and the effective process times.

In order to meet the first research objective, which is introduced in Chapter 1, the derived control framework is tested. It is proved that the framework itself works properly, but the EPT measurement is wrong because it is dependent on the capacity utilization. Therefore the control framework performs poorly for high utilization levels. Also the in Section 1.1 discussed vicious circle is detected for systems with extremely low utilization levels.

The second research objective is met by deriving a new EPT algorithm, which makes use of the authorization times which are determined by a combination of the high and low level controller. Now, the EPTs are no longer dependent on the utilization of the workstations. The control framework performs significantly better with the new EPT measurement for several types of demand and in both highly and extremely lowly utilized manufacturing systems than with the old EPT measurement. Also the described vicious circle is neutralized by this new method.

Chapter 6

Conclusions and Recommendations

This chapter summarizes the most relevant findings of this research project. Besides that, this chapter also contains some recommendations for future research.

6.1 Conclusions

Until now, no satisfactorily working hierarchical model predictive control framework for a manufacturing system has been deployed in a χ-*Matlab* environment. This thesis describes the improvement of an existing two layer hierarchical MPC framework which uses effective process times as capacity constraints. The improvement consists of two parts; the first is the slightly different formulation of the optimization problem and the use of a better, more powerful optimization toolbox for solving this problem. The second part concerns the improvement of the measurement of the effective process times, for a more realistic capacity estimation. The performance of the newly derived framework is analyzed with the use of a simulation model of a simple manufacturing system.

The two main research objectives were defined in Chapter 1. First, the two layer model predictive control framework has to be improved. Secondly, the EPT measurement in casu condition blocking has to be revised, because the capacity estimation, used for the control framework, occurs wrongly at the moment. These two questions are answered in this section.

The Control Framework

Control frameworks can be classified in several ways. Mostly, classification is based on the architecture of the framework. This research makes use of a hierarchical control framework, more specifically a two layer model predictive control framework.

The chosen framework was already presented by Tolboom [Tol04], though some errors occurred during simulation of this framework. Therefore, only the rough line of reasoning is adapted from his work, and the framework is completely 'refurnished'.

The high level control layer consists of a model predictive controller. MPC is a model based advanced control strategy, which makes use of a dynamical model of the process to obtain the control signal by minimizing an objective function. MPC uses this model to generate predictions of the future behavior of the process. Based on these predictions, an objective function is optimized with regard to the future inputs of the process. Although prediction and optimized inputs are computed over a future horizon, only the new values of the inputs for the next sample are actually implemented. This feature is known as the receding horizon strategy.

The model predictive controller, which is used in this research, tries to minimize a cost function, which contains all costs involved in manufacturing the demanded products. Cost parameters are defined for the release (or production) cost per workstation, the wip (or buffer) cost per workstation, stock cost for a surplus of finished products and back order cost for potential backlog. The optimization has to obey several constraints, which can be divided into two sets; the non-linear capacity constraints and the linear mass conservation laws. The latter are fairly intuitive to derive, the first are more difficult and are partially explained here.

The non-linear capacity constraints are represented by the characteristic curves of each workstation. The characteristic curve captures the highly non-linear relationship between throughput and work in process. This curve can be derived using the following variables; the mean effective process time, the squared coefficient of variation of the effective process time and the squared coefficient of variation of the interarrival time. The first two are determined by an EPT algorithm.

Performance of the Control Framework

The above described control framework is tested on a simple discrete event system for performance evaluation of the framework. The used manufacturing system model consists of a generator, two infinitely buffered workstations and an exit process.

Before actual simulation can commence, many assumptions on the discrete event model have to be made, and tuning of the controller parameters has to be performed. Two major assumptions on the model are to cancel out both resource blocking and port blocking. The third type of blocking, condition blocking must be included, since EPT measurements showed difficulties to handle this type of blocking and one of the goals of this research is to solve this problem.
Tuning of the model predictive controller is performed by running some preliminary test runs. The most important tuning parameters are the length of the planning horizon, the start vector of the optimization and the choice of the solver. Another important

factor which increases the speed of the optimization process is the addition of extra constraints to support the non-linear capacity constraints.

The input parameter for the model is the demand, which varies over the time horizon. Process time distributions of both machines are chosen to be equal during simulation runs to ensure a proper comparison between different simulations. Several types of demand and values of capacity utilization are used for performance evaluation. The performance measures, which are used throughout this thesis, are the buffer (or wip) levels of both workstations, the production of the system with respect to the demand and the effective process time of the workstation.

Many experiments are conducted to evaluate the performance of the presented two layer hierarchical model predictive control framework. The most striking results are the fact that the control framework itself performs properly, but EPT measurements cause problems, especially in highly utilized setups. EPTs turn out to be dependent on the capacity utilization, which is wrong per definition.
However, also experiments with extremely low utilization levels perform not well, due to the vicious circle which is initiated by a too low capacity estimation of the controller. The appearance of this phenomenon causes the controller to react slower to demand fluctuations.
The implemented control framework however, works as desired; demands are met while the wip and backlog levels stay reasonably low for lowly utilized manufacturing sytems. This answers in fact the first research objective, to improve the existing framework using slightly different 'furniture' and to prove it functions.

Performance Improvement; introducing a new EPT Measurement Method

Performance improvement of the control framework must be carried out in order to be able to properly control highly utilized manufacturing systems. EPT measurement has to be improved in order to achieve this goal.

Capacity was underestimated by the old EPT measurement method. This was due to the existing condition blocking, which causes products to stay at the machine, while the downstream workstation could in fact already receive the product. When EPT measurement starts at the arrival time of the current product at the current workstation or at the departure of the previous product at the downstream workstation, EPTs are often measured too high. EPT measurement should use the authorization time of the current product at the current workstation instead of the arrival time. The determination of these authorization times must be performed accurately, and is done by the low level controllers, which are present in the buffers of the workstations.

With the newly developed EPT measurement method, again many experiments are conducted. First a repetition of the lowly utilized case is performed and the control framework proves to be much better working. The effective process times are no longer dependent on the capacity utilization and therefore, much better results are obtained.

Both wip levels and backlog levels stay very low in this situation. Also in a highly utilized setup, the control framework works properly. Only with extremely high utilization levels, the wip and backlog rise high again. In extremely lowly utilized systems however, the control framework performs much better, because the vicious circle is completely neutralized by the new method.

The above answers the second research question, which is to review current EPT measurement in case of condition blocking. With this improvement, the performance of the control framework increases significantly.

6.2 Recommendations

Several questions have come forward during this research project that remain unanswered. Based on these questions, recommendations for further research are formulated.

Complexity of the Manufacturing System

The described manufacturing system, which is used throughout this thesis is just a simple line with two infinitely buffered workstations. This system is satisfactory to demonstrate the working of the control framework, but for further evaluation, a more complex system is desired. This complexity can be introduced in various ways. Much of the complexity reseeds in the general assumptions which are made for the model.

Instead of using a simple manufacturing line, a more complex system can be used. An example of such a system is a re-entrant line. This reentrance has two major effects on the control framework. First of all, the controller has to divide the capacity of one workstation over two flows, which is more difficult to control. Secondly, the effective process times are harder to measure, since the authorization times are influenced by the sequencing policies, which can give priority to products of one of the two flows.

Another form of bringing more complexity to the manufacturing system is by introducing different (types of) products. When products are not equal, but have different due dates, the sequencing rules in the low level controller become an important aspect in the calculation of authorization times. This also applies to products, which have different process times on the same machine. So the effective process times can alter due to these sequencing rules, which has an important impact on the high level controller because of the effect of the EPTs on the capacity constraints.

Another factor which can be introduced is the placement of parallel machines in one workstation. Research in the past has shown that it is quite difficult to determine the effective process time in such a case. This is even more hard when condition blocking occurs in combination with these parallel machines.

One final addition within this topic is the introduction of resource or port blocking in combination with the already present condition blocking. This is especially interesting

for EPT measurement, but also for general control, since resource blocking introduces new constraints for the buffer levels of the system.

The general question in all the above remarks is the same; is the presented control framework able to control more complex manufacturing systems? Here lies a great opportunity for further research.

Tuning of the Model Predictive Controller

The model predictive controller which is used in this research is tuned with the aid of several preliminary test runs. The experiments proved the working of the optimization process, but improvement can be carried out at several aspects of the optimization tool.

Improvement can be achieved by stretching the time horizon. In an ideal system, the horizon is infinite. When an infinite horizon is used, stability is guaranteed in the optimal solution. A drawback of this method is that the demand should also be known (or estimated) for the complete horizon. Stability can also be reached at a finite end time by using the steady-state situation as constraint. This means the simulation will stay at that situation with minimal costs, which resembles the infinite horizon situation.

A drawback of the used *Tomlab* solver *conSolve* is that it cannot deal with integer variables. All variables (throughput or wip levels) are treated as reals, which is not possible in the physical version of the used manufacturing system. To overcome this difficulty, throughout this research high throughputs are used. When the desired throughput would for instance fluctuate between 1 and 5 products per planning cycle, the rounding of an optimal throughput which contains a half product becomes very influential. In such cases a (mixed-) integer solver is recommended.

Because the optimization process of the proposed control framework is of a very big importance, the above questions should be considered when continuing with this research.

Performance of the Control Framework

The results of the simulation experiments in this thesis are compared to the results which were obtained by using the old EPT measurement. This comparison indicated that the presented framework works much better with the new EPT measurement. For a better evaluation of the presented framework, it is recommended to compare the performance of this framework to that of several other control approaches.

Bibliography

[Bak01] P.P. van Bakel. Effective process times for batch machines. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, November 2001. SE 420284.

[Bak03] P.P. van Bakel, J.H. Jacobs, L.F.P. Etman, and J.E. Rooda. Quantifying variability of batching equipment using effective process times. *IEEE Transactions on Semiconductor Manufacturing*, 2003. Submitted.

[Bal01] F. Balduzzi. Hybrid system model of production work-cells and its optimal control subject to logical constraints. In *ETFA 8th International Conference on Emerging Technologies and Factory Automation Proceedings*, pages 455–463, IEEE, Piscataway, NJ, USA, October 2001.

[Cam03] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, London, 2nd edition, 2003.

[Dil91] D.M. Dilts, N.P. Boyd, and H.H. Whorms. Nonhierarchical control of manufacturing systems. *Journal of Manufacturing Systems*, 10(1):79–93, 1991.

[Duf86] N.A. Duffie and R.S. Piper. Effectiveness of flexible routing control. *Journal of Manufacturing Systems*, 5(2):137–139, 1986.

[Duf87] N.A. Duffie and R.S. Piper. Non-hierarchical control of a flexible manufacturing cell. *Robotics and Computer-Integrated Manufacturing*, 3(2):175–179, 1987.

[Fow02] J.W. Fowler, G.L. Hogg, and S.J. Mason. Workload control in the semiconductor industry. *Production Planning and Control*, 13(7):568–578, October 2002.

[Ger89] S.B. Gershwin. Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems. In *Proceedings of the IEEE*, volume 77, pages 195–209, January 1989.

[Hac89] S.T. Hackman and R.C. Leachman. A general framework for modeling production. *Management Science*, 35(4):478–495, April 1989.

[Her02] S.S. Heragu, R.J. Graves, B. Kim, and A. St. Onge. Intelligent agent based framework for manufacturing systems control. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, 32(5):560–573, September 2002.

[Hop01] W.J. Hopp and M.L. Spearman. *Factory Physics: foundations of manufacturing management*. McGraw-Hill, London, 2nd edition, 2001.

[Hul62] T.E. Hull and A.R. Dobell. Random number generators. *SIAM Rev.*, (4):230–254, 1962.

[Jac01] J.H. Jacobs, L.F.P. Etman, E.J.J. van Campen, and J.E. Rooda. Quantifying operational time variability: the missing parameter for cycle time reduction. *IEEE/SEMI Advanced semiconductor manufacturing conference*, pages 1–10, 2001.

[Jac03] J.H. Jacobs, L.F.P. Etman, E.J.J. van Campen, and J.E. Rooda. Characterization of operational time variability using effective process times. *IEEE Transactions on Semiconductor Manufacturing*, 16(3):511–520, August 2003.

[Kem03] K.G. Kempf. Intel five-machine six step mini-fab description, October 2003. `http://www.eas.asu.edu/~aar/research/intel/papers/fabspec.html`.

[Khi32] A. Khinchine. Mathematical theory of stationary queues. *Mat. Sbornik*, 39:73–84, 1932.

[Kle01] J.J.T. Kleijn and J.E. Rooda. χ *Manual*. Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Technology, 2001.

[Koc03] A. A. A. Kock. Performance evaluation and simulation meta modeling of single server flow lines subject to blocking: an effective process time approach. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, December 2003. SE 420367.

[Lea01] R.C. Leachman. Semiconductor production planning. In P.M. Pardalos and M.G.C. Resende, editors, *Handbook of Applied Optimization*, pages 746–762, New York, New York, USA, 2001.

[Lin91] G.Y. Lin and J.J. Solberg. Effectiveness of flexible routing control. *International Journal of Flexible Manufacturing Systems*, 3:189–211, 1991.

[Pol30] F. Pollaczek. Uber eine aufgabe der wahrscheinlichkeitstheorie. *Mathematische Zeitschrift*, 32:64–100 and 729–750, 1930.

[Roo02] M. Rooney. Effective process times characterization for workstations with unequal machines, exceptional first lots and idling. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, July 2002. SE 420306.

[Sat96] L. Sattler. Using quequeing curve approximations in a fab to determine productivity improvement. In *IEEE/SEMI 1996 Advanced semiconductor manufacturing conference and workshop*, pages 140–145, IEEE, Cambridge, MA, USA, November 1996.

[Son02] C. Song, C. Gao, H. Wang, and P. Li. A framework of hierarchical receding control policy for production systems. In *Proceedings of the 2002 American Control Conference*, volume 6, pages 5043–5048, American Automatic Control Council, Danvers, MA, USA, May 2002.

[Sou94] J.B. Sousa and F.L. Pereira. A receding horizon strategy for the hierarchical control of manufacturing systems. In *Proceedings of the Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, pages 443–450, IEEE Comput. Soc. Press, Los Alamitos, CA, USA, October 1994.

[Tij94] H.C. Tijms. *Stochastic Models, An algorithmic approach*. John Wiley & Sons, Chichester, 1994.

[Tol04] T. Tolboom. Control of semiconductor wafer fabrication. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, May 2004. SE 420387.

[Var03] F.D. Vargas-Villamil, D.E. Rivera, and K.G. Kempf. A hierarchical approach to production control of re-entrant semiconductor manufacturing lines. *IEEE Transactions on Control Systems Technology*, 11(4):578–587, July 2003.

[Web03] S. Weber. Design of real-time supervisory control systems. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, February 2003. SE 420330.

[Wul02] F.J.J. Wullems. Data collection for simulation of flow lines with blocking; the role of machine failure in throughput loss. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, November 2002. SE 420316.

Appendix A

Optimization Example

The optimization tool, which is used throughout this thesis, is more extensively worked out in this appendix. First, the relevant equations of the optimization tool of Section 4.4 are provided again. In the second part of this appendix, these equations will be elaborated for a fictional optimization run of three periods.

A.1 Optimization Model

Below the objective function, or cost function, is provided again in (A.1) with all boundaries for the variables (A.2a)–(A.2d) and all constraints, both non-linear (A.3) and linear (A.4)–(A.5). Note that no new equations are introduced, only a clear survey of the total optimization problem, which is derived in Section 4.4, is provided.

$$\min_{x(t+i|t)} \sum_{i=0}^{p-1} c_1^T \cdot \underline{\mathrm{x}}(t+i|t) + c_2^T \cdot \underline{\mathrm{w}}(t+i+1|t) + c_3 \cdot y(t+i+1|t) + c_4 \cdot z(t+i+1|t) \quad \text{(A.1)}$$

subject to:

$$0 \leq x_0(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1 \qquad \text{(A.2a)}$$

$$0 \leq w_k(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1,\ k = \{1, 2\} \qquad \text{(A.2b)}$$

$$0 \leq y(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1 \qquad \text{(A.2c)}$$

$$0 \leq z(t+i|t) \qquad \forall t,\ i = 0, \ldots, p-1 \qquad \text{(A.2d)}$$

$$\left((c_{a,k,i}^2 + c_{e,k,i}^2) \cdot t_{e,k,i}^2 \delta_{k,i} + 2t_{e,k,i} \cdot (1 - t_{e,k,i}\delta_{k,i})\right) \cdot \delta_{k,i} - 2w_{k,i}^* \cdot (1 - t_{e,k,i}\delta_{k,i}) \leq 0$$
$$i = 0, \ldots, p-1,\ k = \{1, 2\} \quad \text{(A.3)}$$

$$w_k(t+i+1|t) = w_k(t+i|t) + x_{k-1}(t+i|t) - x_k(t+i|t) \qquad \forall t,\ i = 0,\dots,p-1,\ k = \{1,2\} \quad \text{(A.4)}$$

$$y(t+i+1|t) - z(t+i+1|t) = y(t+i|t) - z(t+i|t) + x_2(t+i|t) - d(t+i|t) \qquad \forall t,\ i = 1,\dots,p-1 \quad \text{(A.5)}$$

A.2 Three Period Evaluation of the Optimization Model

With the described manufacturing line of Figure 4.1, that means a generator, two infinitely buffered workstations and an exit process (so $k = 1,2,3$) and a planning horizon of three periods (so $p = 3$ and $i = 0,1,2$), an example is given here how to use all equations that are provided in the previous section. To evaluate these functions, six quantities have to be measured in the χ-model, which function as input parameters: $w_1(t)$, $w_2(t)$, $w_3(t)$, $d(t)$, $d(t+1)$ and $d(t+2)$.

Constraints for the first planning period, $i = 0$:

$$\begin{aligned}
&\left((c_{a,1,0}^2 + c_{e,1,0}^2)\cdot t_{e,1,0}^2 \cdot x_1(t|t) + 2t_{e,1,0}\cdot(1 - t_{e,1,0}\cdot x_1(t|t))\right)\cdot x_1(t|t) \\
&\qquad - 2w_1(t+1|t)\cdot(1 - t_{e,1,0}\cdot x_1(t|t)) \le 0 \\
&\left((c_{a,2,0}^2 + c_{e,2,0}^2)\cdot t_{e,2,0}^2 \cdot x_2(t|t) + 2t_{e,2,0}\cdot(1 - t_{e,2,0}\cdot x_2(t|t))\right)\cdot x_2(t|t) \\
&\qquad - 2w_2(t+1|t)\cdot(1 - t_{e,2,0}\cdot x_2(t|t)) \le 0 \\
&w_1(t+1|t) = w_1(t|t) + x_0(t|t) - x_1(t|t) \\
&w_2(t+1|t) = w_2(t|t) + x_1(t|t) - x_2(t|t) \\
&y(t+1|t) - z(t+1|t) = y(t|t) - z(t|t) + x_2(t|t) - d(t|t)
\end{aligned}$$

Constraints for the second planning period, $i = 1$:

$$\begin{aligned}
&\left((c_{a,1,1}^2 + c_{e,1,1}^2)\cdot t_{e,1,1}^2 \cdot x_1(t+1|t+1) + 2t_{e,1,1}\cdot(1 - t_{e,1,1}\cdot x_1(t+1|t+1))\right)\cdot x_1(t+1|t+1) \\
&\qquad - 2w_1(t+2|t+1)\cdot(1 - t_{e,1,1}\cdot x_1(t+1|t+1)) \le 0 \\
&\left((c_{a,2,1}^2 + c_{e,2,1}^2)\cdot t_{e,2,1}^2 \cdot x_2(t+1|t+1) + 2t_{e,2,1}\cdot(1 - t_{e,2,1}\cdot x_2(t+1|t+1))\right)\cdot x_2(t+1|t+1) \\
&\qquad - 2w_2(t+2|t+1)\cdot(1 - t_{e,2,1}\cdot x_2(t+1|t+1)) \le 0 \\
&w_1(t+2|t+1) = w_1(t+1|t+1) + x_0(t+1|t+1) - x_1(t+1|t+1) \\
&w_2(t+2|t+1) = w_2(t+1|t+1) + x_1(t+1|t+1) - x_2(t+1|t+1) \\
&y(t+2|t+1) - z(t+2|t+1) = y(t+1|t+1) - z(t+1|t+1) + x_2(t+1|t+1) \\
&\qquad - d(t+1|t+1)
\end{aligned}$$

Constraints for the third planning period, $i = 2$:

$$\begin{aligned}\big((c_{a,1,2}^2+c_{e,1,2}^2)\cdot t_{e,1,2}^2\cdot x_1(t+2|t+2)+2t_{e,1,2}\cdot(1-t_{e,1,2}\cdot x_1(t+2|t+2))\big)\cdot x_1(t+2|t+2)\\ -2w_1(t+3|t+2)\cdot(1-t_{e,1,2}\cdot x_1(t+2|t+2))\leq 0\end{aligned}$$

$$\begin{aligned}\big((c_{a,2,2}^2+c_{e,2,2}^2)\cdot t_{e,2,2}^2\cdot x_2(t+2|t+2)+2t_{e,2,2}\cdot(1-t_{e,2,2}\cdot x_2(t+2|t+2))\big)\cdot x_2(t+2|t+2)\\ -2w_2(t+3|t+2)\cdot(1-t_{e,2,2}\cdot x_2(t+2|t+2))\leq 0\end{aligned}$$

$$w_1(t+3|t+2) = w_1(t+2|t+2) + x_0(t+2|t+2) - x_1(t+2|t+2)$$

$$w_2(t+3|t+2) = w_2(t+2|t+2) + x_1(t+2|t+2) - x_2(t+2|t+2)$$

$$\begin{aligned}y(t+3|t+2) - z(t+3|t+2) = y(t+2|t+2) - z(t+2|t+2) + x_2(t+2|t+2)\\ - d(t+2|t+2)\end{aligned}$$

When all optimization variables for these three periods are put together, the vector $\underline{x}$ with 21 rows arises:

$$\begin{aligned}\underline{x} = [&x_0(t|t),\ x_1(t|t),\ x_2(t|t),\ w_1(t+1|t),\ w_2(t+1|t),\ y(t+1|t),\ z(t+1|t),\\ &x_0(t+1|t+1),\ x_1(t+1|t+1),\ x_2(t+1|t+1),\ w_1(t+2|t+1),\ w_2(t+2|t+1),\\ &y(t+2|t+1),\ z(t+2|t+1),\ x_0(t+2|t+2),\ x_1(t+2|t+2),\ x_2(t+2|t+2),\\ &w_1(t+3|t+2),\ w_2(t+3|t+2),\ y(t+3|t+2),\ z(t+3|t+2)]^T\end{aligned}$$

Only the first three variables are actually used as input for the χ-model, these are $x_0(t|t)$, $x_1(t|t)$ and $x_2(t|t)$.

One can see that with only two workstations and three periods, the number of equations and variables is fairly high. But in comparison with other methods, the number of equations is still limited, since this problem grows linearly, where other methods often grow exponentially.
Still, it is of great importance to work very secure when actually implementing this problem. In practice, the horizon will be greater than three, so even in this research project the number of equations and variables is larger than presented in this appendix.

Appendix B

Simulation Framework

The simulation framework which is used for the experiments of Chapter 5 is programmed with use of the following programs; the specification language χ, the scripting language *Python* and the mathematical program *Matlab* with the external toolbox *Tomlab*.

The discrete event simulation model is modeled in χ 0.8. The demand for lots must be variable per simulation run, therefore a special demand generator is developed in *Matlab*. EPT calculations, that is the calculation of the t_e's, c_e^2's and c_a^2's, are performed by *Matlab*. For the non-linear optimization, an external toolbox of *Matlab* is used, called *Tomlab*. Communication between these several blocks of the simulation framework is realized through two *Python* scripts. The simulation framework is depicted in Figure B.1.

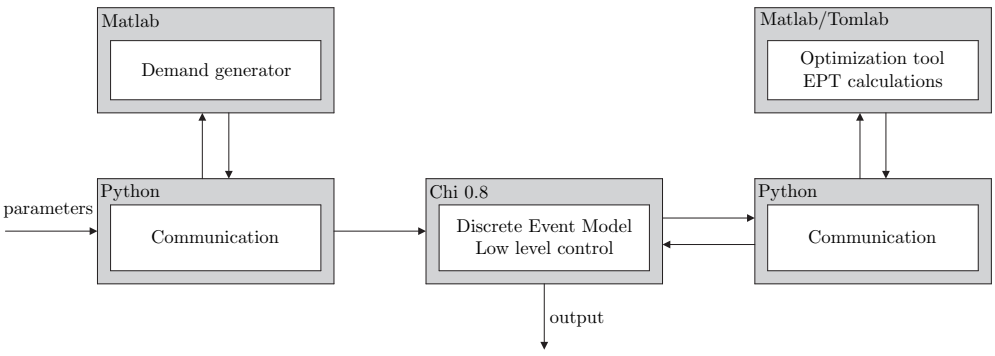


Figure B.1: Simulation framework.

Before the actual program files are given in this appendix, first the simulation framework of Figure B.1 is converted into all files, which are used for the simulation. When the structure is clear, the actual code of these files is given. Figure B.2 shows all files and their mutual communications.

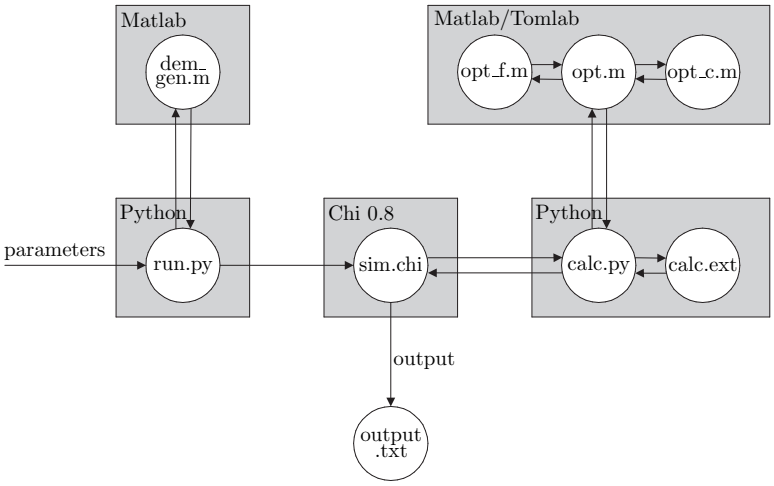


Figure B.2: Simulation framework with all used files.

The next sections show all files per program. Some explanation is provided when the working of the files is not clear.

B.1 Matlab

This section contains the demand generator in Matlab. The optimization files are displayed in Section B.4. The three types of demand, which are shown in Figure 5.2, can be generated with this file.

dem_gen.m

```
function [dem] = dem_gen(n, Dtype, Dper, Dmin, Dmax)

% n         [nat]       horizon
% Dtype     [string]    demand function type (rampup, rampdown, sinus)
% Dmin      [real]      minimum value of demand
% Dmax      [real]      maximum value of demand
% Dper      [real]      (transition) period length

if strcmp(Dtype, 'rampup')
    for i = 1:Dper
        A(i,1) = Dmin + i/Dper*(Dmax-Dmin);
    end
    B = Dmax*ones(n-Dper,1);
    dem = [A; B];

elseif strcmp(Dtype, 'rampdown')
    for i = 1:Dper
        A(i,1) = Dmax + i/Dper*(Dmin-Dmax);
    end
    B = Dmin*ones((n-Dper),1);
```

```
    dem = [A; B];

elseif strcmp(Dtype, 'sinus')
    A = (Dmax - Dmin)/2;
    B = (2*pi/Dper);
    C = mean([Dmin,Dmax]);
    j = 0;
    for i = 1:n
        dem(i,1) = A*sin(B*j) + C;
        j=j+1;
    end
end
```

B.2 Python

This section shows both *Python* files (**run.py** and **calc.py**) which are used in the simulation framework, as well as the file **calc.ext** which contains the list of external specifications for the χ-compiler.

run.py

```
#!/usr/bin/python
import os, string, math, sys, pymat

input = sys.argv[1]
I = pymat.open()
pymat.eval(I, 'dem = dem_gen '+str(input))
dem = list(pymat.get(I, 'dem'))

i=0
demand=[]
while i < len(dem):
    demand.append(int(dem[i]))
    i=i+1

demand = str(demand)
demand = string.replace(demand,',','')
demand = '\'' + demand + '\''

a='./sim '+str(demand)
J = os.popen(a)
```

calc.py

```
from Numeric import *
import pymat, os

def openTomlab():
  a=os.getcwd()
  global H
  H=pymat.open("cd /usr/home/bdj/bas/tomlab; matlab")
  pymat.eval(H,"cd "+str(a))
  # pymat.eval(H,"diary on")
```

```
  return 1

def closeTomlab():
  pymat.close(H)
  return 1

def mean(a):
  x=array(a)
  pymat.put(H,'x',x)
  pymat.eval(H,'y=mean(x)')
  y=pymat.get(H,'y')
  return y[0]

def scv(a):
  x=array(a)
  pymat.put(H,'x',x)
  pymat.eval(H,'y=var(x)/mean(x)^2')
  y=pymat.get(H,'y')
  return y[0]

def opt(a, b, c, d):
  tccw1=array([a])
  tccw2=array([b])
  wip3=array([c])
  demand=array([d])
  pymat.eval(H,"global tccw1")
  pymat.eval(H,"global tccw2")
  pymat.put(H,'tccw1',tccw1)
  pymat.put(H,'tccw2',tccw2)
  pymat.put(H,'wip3',wip3)
  pymat.put(H,'demand',demand)
  pymat.eval(H,"opt")
  xk=pymat.get(H,'xk')
  # xk                        is stored as [ 10.]  , where xk = 10.0000
  # tuple(xk)                 is stored as (9.999999998967402,)
  # tuple(xk)[0]              is stored as 9.99999999897
  # round(tuple(xk)[0])       is stored as 10.0
  # int(round(tuple(xk)[0])) is stored as 10
  return ( int(round(tuple(xk)[0])), int(round(tuple(xk)[1])), int(round(tuple(xk)[2])) )
```

calc.ext

```
language "python"
file "calc"

ext openTomlab()  -> bool
ext closeTomlab() -> bool
ext mean(n:real*) -> real
ext scv(n:real*)  -> real
ext opt(tccw1:real#real#real#int, tccw2:real#real#real#int, wip3:int, demand:int*) -> int#int#int
```

B.3 Chi

This section contains the whole χ-model which is used for the simulations. Figure 5.1 already showed the general layout of the model, but Figure B.3 also shows the names of the channels, which are used in the model.

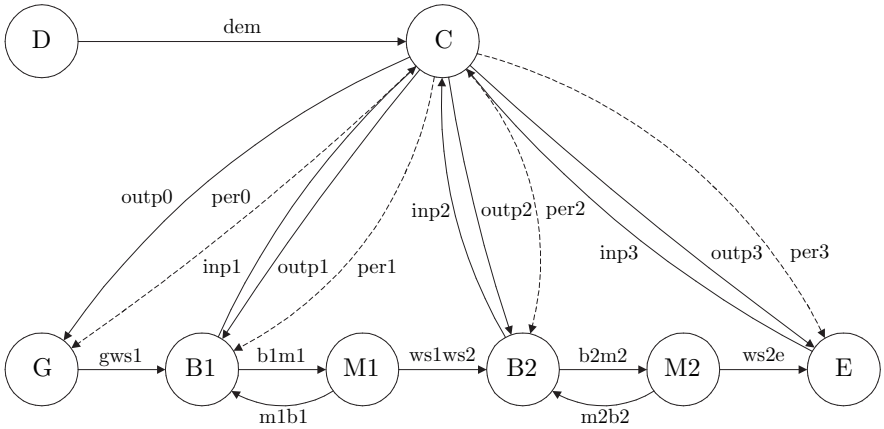


Figure B.3: Simulation scheme in χ.

No further explanation of the χ-files is provided here, since it does not concern a very complicated model. Besides that, the important decisions and assumptions on the model are already worked out in Chapter 4 and Chapter 5.

sim.chi

```
from std import *
from random import *
from fileio import *
from calc import *


type lot  = nat#real#real#real#real#real#real // lotnr # auth1 # fin1 # auth2 # fin2 # ept1 # ept2
   , tcc  = real#real#real                    // te # ce2 # ca2
   , tccw = real#real#real#int                // te # ce2 # ca2 # wip


proc G(gws1:!lot, outp0:?int, per0:?void) =
|[ rq0: int, ta: real, prod: lot, n: nat, t: -> real
 | ta:= 0.0; n:= 1; t:= uniform(0.0,0.05)
 ; *[ true; per0?
      -> outp0?rq0
       ; *[ rq0 > 0
            -> prod.0:= n
             ; gws1!prod
             ; n:= n + 1
             ; rq0:= rq0 - 1
          ]
    ]
]|


proc B1(gws1:?lot, b1m1:!lot, m1b1:?lot, inp1:!tccw, outp1:?int, per1:?void) =
|[ incprods, authprods, outbprods: lot*, prod, tempprod: lot, tcc1: tcc, wip1: int, tccw1: tccw
 , rq1: int, eptcoll, rtcoll: real*, empmch: bool, ta, lastauth: real, n: int
 | incprods:= []; authprods:= []; outbprods:= []; eptcoll:= []; wip1:= +0; empmch:= true; ta:= 0.0
 ; lastauth:= 0.0; n:= +0
```

```
 ; *[ true; gws1?prod
      -> wip1:= wip1 + 1
       ; incprods:= insertandsort(prod,incprods)
       ; [ len(incprods) > 0 and n > 0
           -> tempprod:= hd(incprods)
            ; incprods:= tl(incprods)
            ; tempprod.5:= time
            ; [ n = rq1 -> tempprod.1:= time
              | n < rq1 -> tempprod.1:= lastauth + ta
              ]
            ; lastauth:= tempprod.1
            ; n:= n-1
            ; authprods:= authprods ++ [tempprod]
         | len(incprods) = 0 or n = 0
           -> skip
         ]
    | len(authprods) > 0; delta hd(authprods).1-time
      -> outbprods:= outbprods ++ [hd(authprods)]
       ; authprods:= tl(authprods)
    | len(outbprods) > 0 and empmch; b1m1!hd(outbprods)
      -> outbprods:= tl(outbprods)
       ; empmch:= false
    | true; m1b1?prod
      -> eptcoll:= eptcoll ++ [prod.2 - prod.1]
       ; rtcoll:= rtcoll ++ [prod.1]
       ; wip1:= wip1 - 1
       ; empmch:= true
    | true; per1?
      -> [ len(eptcoll) > 0 and len(rtcoll) > 0 -> tcc1:= tccalg(eptcoll, rtcoll)
         | len(eptcoll) = 0 or len(rtcoll) = 0  -> tcc1:= <0.0, 0.0, 0.0>
         ]
       ; tccw1.0:=tcc1.0; tccw1.1:=tcc1.1; tccw1.2:=tcc1.2; tccw1.3:=wip1
       ; inp1!tccw1
       ; eptcoll:= []; rtcoll:= []
       ; outp1?rq1
       ; n:= rq1
       ; ta:= 24.0 / i2r(rq1)
       ; incprods:= incprods ++ authprods ++ outbprods
       ; authprods:= []; outbprods:= []
       ; *[ len(incprods) > 0 and n > 0
            -> tempprod:= hd(incprods)
             ; incprods:= tl(incprods)
             ; [ n = rq1 -> tempprod.1:= time
               | n < rq1 -> tempprod.1:= lastauth + ta
               ]
             ; lastauth:= tempprod.1
             ; n:= n-1
             ; authprods:= authprods ++ [tempprod]
          ]
    ]
]|


proc M1(b1m1:?lot, m1b1:!lot, ws1ws2:!lot, seed:nat) =
|[ prod: lot, t: -> real
 | t:= uniform(0.20,0.22)
 ; setseed(t,seed)
 ; *[ true; b1m1?prod -> delta sample t; prod.2:=time; m1b1!prod; ws1ws2!prod ]
]|


proc B2(ws1ws2:?lot, b2m2:!lot, m2b2:?lot, inp2:!tccw, outp2:?int, per2:?void) =
```

```
|[ incprods, authprods, outbprods: lot*, prod, tempprod: lot, tcc2: tcc, wip2: int, tccw2: tccw
 , rq2: int, eptcoll, rtcoll: real*, empmch: bool, ta, lastauth, authtime: real, n: int
 | incprods:= []; authprods:= []; outbprods:= []; eptcoll:= []; wip2:= +0; empmch:= true; ta:= 0.0
 ; lastauth:= 0.0; n:= +0
 ; *[ true; ws1ws2?prod
      -> wip2:= wip2 + 1
       ; incprods:= insertandsort(prod,incprods)
       ; [ len(incprods) > 0 and n > 0
           -> tempprod:= hd(incprods)
            ; incprods:= tl(incprods)
            ; tempprod.6:= time
            ; [ n = rq2 -> tempprod.3:= time
              | n < rq2 -> authtime:= lastauth + ta
                           ; [ authtime >= time -> tempprod.3:= authtime
                             | authtime < time  -> tempprod.3:= time
                             ]
              ]
            ; lastauth:= tempprod.3
            ; n:= n-1
            ; authprods:= authprods ++ [tempprod]
         | len(incprods) = 0 or n = 0
           -> skip
         ]
    | len(authprods) > 0; delta hd(authprods).3-time
      -> outbprods:= outbprods ++ [hd(authprods)]
       ; authprods:= tl(authprods)
    | len(outbprods) > 0 and empmch; b2m2!hd(outbprods)
      -> outbprods:= tl(outbprods)
       ; empmch:= false
    | true; m2b2?prod
      -> eptcoll:= eptcoll ++ [prod.4 - prod.3]
       ; rtcoll:= rtcoll ++ [prod.3]
       ; wip2:= wip2 - 1
       ; empmch:= true
    | true; per2?
      -> [ len(eptcoll) > 0 and len(rtcoll) > 0 -> tcc2:= tccalg(eptcoll, rtcoll)
         | len(eptcoll) = 0 or len(rtcoll) = 0  -> tcc2:= <0.0, 0.0, 0.0>
         ]
       ; tccw2.0:=tcc2.0; tccw2.1:=tcc2.1; tccw2.2:=tcc2.2; tccw2.3:=wip2
       ; inp2!tccw2
       ; eptcoll:= []; rtcoll:= []
       ; outp2?rq2; n:= rq2; ta:= 24.0 / i2r(rq2)
       ; incprods:= incprods ++ authprods ++ outbprods
       ; authprods:= []; outbprods:= []
       ; *[ len(incprods) > 0 and n > 0
            -> tempprod:= hd(incprods)
             ; incprods:= tl(incprods)
             ; [ n = rq2 -> tempprod.3:= time
               | n < rq2 -> tempprod.3:= lastauth + ta
               ]
             ; lastauth:= tempprod.3
             ; n:= n-1
             ; authprods:= authprods ++ [tempprod]
         ]
    ]
]|


proc M2(b2m2:?lot, m2b2:!lot, ws2e:!lot, seed:nat) =
|[ prod: lot, t: -> real
 | t:= uniform(0.22,0.24)
 ; setseed(t,seed)
```

```
 ; *[ true; b2m2?prod -> delta sample t; prod.4:=time; m2b2!prod; ws2e!prod ]
]|


func insertandsort(prod:lot,incprods:lot*) -> lot* =
|[ incprods:= incprods ++ [prod]    // a sorting algorithm can be inserted here
 ; ret incprods
]|


func tccalg(eptcoll, rtcoll:real*) -> tcc =
|[ te, ce2, ca2: real
 | te:= mean(eptcoll)
 ; ce2:= scv(eptcoll)
 ; ca2:= scv(rtcoll)
 ; ret <te, ce2, ca2>
]|


proc C(per0, per1, per2, per3:!void, inp3:?int, inp1, inp2:?tccw
      , outp0, outp1, outp2, outp3:!int, dem:?int*, f:!file
      ) =
|[ dummy: bool, wip3, rq0, rq1, rq2: int, tccw1, tccw2: tccw, demand: int*, rqs: int#int#int
 | dummy:= openTomlab()
 ; *[ true
      -> per0!
       ; per1!
       ; per2!
       ; per3!
       ; inp1?tccw1
       ; inp2?tccw2
       ; inp3?wip3
       ; dem?demand
       ; [ len(demand) <  5 -> terminate
         | len(demand) >= 5 -> skip
         ]
       ; [ tccw1.0 = 0.0
           -> rq0:= hd(demand); rq1:= rq0; rq2:= rq0
         | tccw1.0 > 0.0
           -> rqs:= opt(tccw1, tccw2, wip3, demand)
            ; rq0:= rqs.0; rq1:= rqs.1; rq2:= rqs.2
         ]
       ; f! time, "\t", tccw1.0,  "\t", tccw1.1, "\t", tccw1.2, "\t", tccw1.3, "\t", tccw2.0
       , "\t", tccw2.1, "\t", tccw2.2, "\t", tccw2.3, "\t", wip3, "\t", rq0, "\t", rq1
       , "\t", rq2, "\t", hd(demand), "\n"
       ; outp0!rq0
       ; outp1!rq1
       ; outp2!rq2
       ; outp3!hd(demand)
       ; delta 24
    ]
; dummy:= closeTomlab()
]|


proc D(dem:!int*, demand:int*) =
|[ *[ true -> dem!demand; demand:= tl(demand) ] ]|


proc E(ws2e:?lot, inp3:!int, outp3:?int, per3:?void) =
|[ prod: lot, wip3: int, demand: int
 | wip3:= +0
```

```
 ; *[ true; ws2e?prod
      -> wip3:= wip3 + 1
    | true; per3?
      -> wip3:= wip3 - demand
       ; inp3!wip3
       ; outp3?demand
    ]
]|


clus S(seed: nat, demand: int*)=
|[ gws1, b1m1, m1b1, ws1ws2, b2m2, m2b2, ws2e:-lot, outp0, outp1, outp2, outp3, inp3:-int
 , inp1, inp2:-tccw, per0, per1, per2, per3:-void, dem:-int*
 | G(gws1, outp0, per0)
|| B1(gws1, b1m1, m1b1, inp1, outp1, per1)
|| M1(b1m1, m1b1, ws1ws2, seed)
|| B2(ws1ws2, b2m2, m2b2, inp2, outp2, per2)
|| M2(b2m2, m2b2, ws2e, seed)
|| D(dem, demand)
|| E(ws2e, inp3, outp3, per3)
|| C(per0, per1, per2, per3, inp3, inp1, inp2, outp0, outp1, outp2, outp3, dem, fileout("output.txt"))
]|


xper(seed:nat, demand:int*) = |[ S(seed, demand) ]|
```

B.4 Tomlab

The three *Tomlab* files which are used for optimization are provided here. Because the Tomlab package is not a standard toolbox of Matlab, some explanation about the files is provided. This is printed below each *m*-file. Note that much information is already embedded in the files, all text behind a %-symbol denotes a remark.

opt.m

```
warning off MATLAB:divideByZero

global tccw1
global tccw2

te1  = tccw1(1,1) ;  te2  = tccw2(1,1) ;
ce21 = tccw1(1,2) ;  ce22 = tccw2(1,2) ;
ca21 = tccw1(1,3) ;  ca22 = tccw2(1,3) ;
w1   = tccw1(1,4) ;  w2   = tccw2(1,4) ;

w3   = wip3       ; d1   = demand(1)   ;
d2   = demand(2)  ; d3   = demand(3)   ;
d4   = demand(4)  ; d5   = demand(5)   ;

a = -24/te1       ; b = -24/te2        ;
c = 24/te1        ; d = 24/te2         ;

f           = 'opt_f';                                    % Function value
g           = [];                                         % Gradient vector
H           = [];                                         % Hessian matrix
```

```
HessPattern = [];                                                   % Hessian Pattern
x_L         = [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]';
                                                                    % Lower bound(s) of design variables x
x_U         = [ Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf ...
                Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf ]';
                                                                    % Upper bound(s) of design variables x
Name        = [];                                                   % Problem name
x_0         = [ d1 d1 d1 0 0 0 0 d2 d2 d2 0 0 0 0 d3 d3 d3 0 0 0 0 d4 d4 d4 0 0 0 0 d5 d5 d5 0 0 0 0 ];
                                                                    % Starting values for the optimization
pSepFunc    = [];                                                   % Number of partially separable functions
fLowBnd     = [];                                                   % Lower bound on function

% Linear constraints (A) and lower and upperbounds on them (b_L and b_U)
A = [ -1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0 -1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0 -1  0  0  1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0 -1  0  0  0 -1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0 -1  0  0  0 -1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0 -1  1  0  0 -1  0  0  1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0 -1  1  0  0 -1  0  0  1 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  1  0  1  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  1  0  1  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  1  0  0 -1  0  0  1 -1  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  1  0  1  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  1  0  1  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  1  0  0 -1  0  0  1 -1 ;
       0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 ;
       0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 ;
       0  1  0  a  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  1  0  a  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  a  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  a  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  a  0  0  0 ;
       0  0  1  0  b  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  1  0  b  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  b  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  b  0  0  0  0  0  0  0  0  0 ;
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  b  0  0 ];
b_L = [ w1 w2 w3-d1 0 0 -d2 0 0 -d3 0 0 -d4 0 0 -d5 0 0 0 0 0 0 0 0 0 0 ...
        -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf ];
b_U = [ w1 w2 w3-d1 0 0 -d2 0 0 -d3 0 0 -d4 0 0 -d5 c c c c c d d d d d ...
          0    0    0    0    0    0    0    0    0    0  ];

c           = 'opt_c';                                              % Nonlinear constraint vector
dc          = [];                                                   % Nonlinear constraint gradient matrix
d2c         = [];                                                   % The second part of the Hessian
ConsPattern = [];                                                   % Constraint Pattern
c_L         = [ -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf ]; % Lower bound(s) on nonlinear constraint(s)
c_U         = [ 0 0 0 0 0 0 0 0 0 0 ] ;                             % Upper bound(s) on nonlinear constraint(s)
x_min       = [];                                                   % Plot region parameters (optional)
x_max       = [];                                                   % Plot region parameters (optional)
x_opt       = [];                                                   % Known optimal point (optional)
```

```
f_opt       = [];                                                    % Known optimal function value (optional)

% Assign routine for defining the problem
Prob = conAssign(f, g, H, HessPattern, x_L, x_U, Name, x_0, pSepFunc, fLowBnd,...
                 A, b_L, b_U, c, dc, d2c, ConsPattern, c_L, c_U,...
                 x_min, x_max, f_opt, x_opt);

% Calling driver routine tomRun to run the solver
Result = tomRun('conSolve', Prob, [], 2);

dateandtime = fix(clock)                                            % Displays date and time
xk=Result.x_k                                                       % Result.x_k: optimal decision variable
fk=Result.f_k                                                       % Result.f_k: optimal value
```

Matrix A, which consists of 35x35 elements needs some extra attention. It contains all linear constraints of the optimization problem, each row is a constraint. The lower and upper bounds which belong to this matrix are provided in the vectors b_L and b_U, respectively.

The first fifteen rows denote the equality constraints of the problem, three per period. These three are represented by equations (A.4) with $k = 1, 2$ and (A.5). Because these rows are equality constraints, the same value is filled in vector b_L and b_U.

Rows 16–20 contain the extra linear constraints which represent the maximum capacity of workstation 1, whereas rows 21–25 represent the maximum capacity of workstation 2. These additional constraints are introduced in Section 5.2, under the subsection 'Tuning Parameters'.

Rows 26–30 and 31–35 represent the capacity slopes in the origin of the characteristic curves of workstation 1 and 2, respectively.

opt_f.m

```
function f = opt_f(x, Prob)

c1 = [ 0.5 0.5 0.5 ] ;  % release cost
c2 = [ 1 2 ]         ;  % wip cost
c3 = 5               ;  % inventory holding cost
c4 = 10              ;  % backorder cost

f = c1(1)*x(1) + c1(2)*x(2) + c1(3)*x(3) + c2(1)*x(4) + c2(2)*x(5) + c3*x(6) + c4*x(7) + ...
    c1(1)*x(8) + c1(2)*x(9) + c1(3)*x(10) + c2(1)*x(11) + c2(2)*x(12) + c3*x(13) + c4*x(14) + ...
    c1(1)*x(15) + c1(2)*x(16) + c1(3)*x(17) + c2(1)*x(18) + c2(2)*x(19) + c3*x(20) + c4*x(21) + ...
    c1(1)*x(22) + c1(2)*x(23) + c1(3)*x(24) + c2(1)*x(25) + c2(2)*x(26) + c3*x(27) + c4*x(28) + ...
    c1(1)*x(29) + c1(2)*x(30) + c1(3)*x(31) + c2(1)*x(32) + c2(2)*x(33) + c3*x(34) + c4*x(35) ;
```

This m-file contains the objective or cost function with its 35 elements. Also the values of the four cost parameters $\underline{c}_1$, $\underline{c}_2$, c_3 and c_4 are declared.

opt_c.m

```
function c = opt_c(x, Prob)

global tccw1
global tccw2

te1  = tccw1(1,1) ; te2  = tccw2(1,1) ;
ce21 = tccw1(1,2) ; ce22 = tccw2(1,2) ;
ca21 = tccw1(1,3) ; ca22 = tccw2(1,3) ;

% Non-linear constraints for workstation 1
c(1) = ( (ce21+ca21) * (te1*te1*x(2)) + 2*te1*(24-te1*x(2)) ) * x(2) - 12*x(4)*(24-te1*x(2));
c(2) = ( (ce21+ca21) * (te1*te1*x(9)) + 2*te1*(24-te1*x(9)) ) * x(9) - 12*x(11)*(24-te1*x(9));
c(3) = ( (ce21+ca21) * (te1*te1*x(16)) + 2*te1*(24-te1*x(16)) ) * x(16) - 12*x(18)*(24-te1*x(16));
c(4) = ( (ce21+ca21) * (te1*te1*x(23)) + 2*te1*(24-te1*x(23)) ) * x(23) - 12*x(25)*(24-te1*x(23));
c(5) = ( (ce21+ca21) * (te1*te1*x(30)) + 2*te1*(24-te1*x(30)) ) * x(30) - 12*x(32)*(24-te1*x(30));

% Non-linear constraints for workstation 2
c(6) = ( (ce22+ca22) * (te2*te2*x(3)) + 2*te2*(24-te2*x(3)) ) * x(3) - 12*x(5)*(24-te2*x(3));
c(7) = ( (ce22+ca22) * (te2*te2*x(10)) + 2*te2*(24-te2*x(10)) ) * x(10) - 12*x(12)*(24-te2*x(10));
c(8) = ( (ce22+ca22) * (te2*te2*x(17)) + 2*te2*(24-te2*x(17)) ) * x(17) - 12*x(19)*(24-te2*x(17));
c(9) = ( (ce22+ca22) * (te2*te2*x(24)) + 2*te2*(24-te2*x(24)) ) * x(24) - 12*x(26)*(24-te2*x(24));
c(10) = ( (ce22+ca22) * (te2*te2*x(31)) + 2*te2*(24-te2*x(31)) ) * x(31) - 12*x(33)*(24-te2*x(31));
```

This m-file contains all ten non-linear constraints. The boundaries are declared in the file **opt.m**, in the vectors c_U and c_L.

Appendix C

Tomlab Output

This appendix contains two sample outputs of the optimization toolbox *Tomlab*. While simulating the experiments, sometimes the diary of the *Matlab*-events is saved to disk for later analysis of the optimization results. As explained in Experiment IV of Chapter 5, sometimes the optimization toolbox is unable to find a feasible optimum. The current vector x_k, which the toolbox was evaluating at that time, is provided instead.

During the simulations of this research project, three different optimization faults occasionally occurred. These are indicated by the following names; 'too high penalty values', 'maximal number of iterations' and 'problem is maybe infeasible'. These problems occur at different circumstances, with different values of the measured wip-levels and different future demands. Most of the time, the problem is actually infeasible, due to absurd demands and utilizations. This can happen at the top of a sine curve, but because the demand always decreases after such a top, the controller can restore the backlog which is created due to infeasibility of the optimization problem.

Output examples of the three occurring faults are provided below.

Too high penalty values

```
===== * * * ================================================================ * * *
TOMLAB /SOL - Time limited demonstration single user license          Valid to 2005-09-28
================================================================================
Problem: No Init File    -  1: User Problem 1   f_k     856.127701686405089276
                                        sum(|constr|)      0.556172459277683995
                                f(x_k) + sum(|constr|)    856.683874145682807466
                                             f(x_0)    763.500000000000000000

Solver: conSolve.  EXIT=1.  INFORM=106.
Schittkowski SQP algorithm with BFGS update
Too high penalty values

FuncEv 29340 GradEv  815 ConstrEv 29340 Iter  218
CPU time: 22.830000 sec. Elapsed time: 22.833445 sec.
Starting vector x (Length 35. Print first 20):
```

```
x_0: 101.000000 101.000000 101.000000   0.000000   0.000000   0.000000
       0.000000 101.000000 101.000000 101.000000   0.000000   0.000000
       0.000000   0.000000 102.000000 102.000000 102.000000   0.000000
       0.000000   0.000000
Optimal vector x (Length 35. Print first 20):
x_k:  64.056683 100.023112 103.247413   4.033570  10.775699   0.000000
       0.752587  97.589022  97.760096 101.752587   3.862496   6.783208
       0.000000   0.000000 102.717804 102.327356 102.000000   4.252944
       7.110564   0.000000
Lagrange multipliers v. Vector length 80 (Print first 20):
v_k:   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   1.499429e+01
       8.156622e-03   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
       4.011243e+00   1.098871e+01   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
       0.000000e+00   4.979258e+00
Gradient g_k Largest abs(gradient)          10.00000007020374149 (Print first 20):
g_k:   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00   2.000000e+00   5.000000e+00
       1.000000e+01   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00   2.000000e+00
       5.000000e+00   1.000000e+01   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e-00
       2.000000e+00   5.000000e+00
cErr  -1.141386e-01  -6.278770e-02  -1.657216e-03  -2.518851e-02  -3.800184e-02  -1.744847e-01
      -2.177123e-02  -5.258016e-13  -8.239565e-02  -3.574702e-02
9 inequalities off more than cTol =       1.000e-06
Worst constraint validation =       1.745e-01 for constraint # 6
Projected gradient gPr:  Largest abs(Projected gradient)           0.00227138479916811 (Print first 20):
gPr:   1.848104e-03   1.706076e-03   2.382691e-04   1.420276e-04   1.467807e-03  -8.881784e-16
      -2.382691e-04  -2.271385e-03  -1.989671e-03  -2.382691e-04  -1.396862e-04  -2.835950e-04
      -3.552714e-15   0.000000e+00   4.533907e-04   2.835950e-04  -1.665335e-15   3.010952e-05
      -2.220446e-15  -2.664535e-15
*** WARNING: 26 reduced gradient values > 1E-5 *** Worst value: 2.271385e-03
    xVarLow 6  xVarLow 13 xVarLow 14 xVarLow 20 xVarLow 21 xVarLow 28
    xVarLow 34 LinEq   1  LinEq   2  LinEq   3  LinEq   4  LinEq   5
    LinEq   6  LinEq   7  LinEq   8  LinEq   9  LinEq   10 LinEq   11
    LinEq   12
v:   1.500024e+01   4.434678e+00   1.056532e+01   4.484910e+00   1.051509e+01   1.500007e+01
     1.499966e+01  -4.981519e-01  -1.079339e+00  -1.391942e+01  -5.022714e-01  -1.074668e+00
    -3.919178e+00  -4.995466e-01  -1.105355e+00  -4.484500e+00  -5.002566e-01  -1.110166e+00
    -4.999590e+00  -5.008952e-01

=== * * * ================================================ * * *


xk =

   64.0567
  100.0231
  103.2474
    4.0336
   10.7757
         0
    0.7526
   97.5890
   97.7601
  101.7526
    3.8625
    6.7832
         0
         0
  102.7178
  102.3274
  102.0000
    4.2529
    7.1106
```

```
         0
         0
  102.6280
  102.5995
  102.2361
    4.2815
    7.4739
    0.2361
         0
  103.5605
  103.4608
  102.6230
    4.3812
    8.3117
         0
    0.1409


fk =

  856.1277
```

Maximal number of iterations reached

```
===== * * * ========================================================================= * * *
TOMLAB /SOL - Time limited demonstration single user license          Valid to 2005-09-28
================================================================================
Problem: No Init File    -  1: User Problem 1   f_k    1271.935688589917162972
                                      sum(|constr|)      8.033410767192242119
                               f(x_k) + sum(|constr|)   1279.969099357109371340
                                           f(x_0)    759.000000000000000000

Solver: conSolve.  EXIT=1.  INFORM=101.
Schittkowski SQP algorithm with BFGS update
Maximal number of iterations reached

FuncEv 62172 GradEv 1727 ConstrEv 62172 Iter  500
CPU time: 49.760000 sec. Elapsed time: 49.862324 sec.
Starting vector x (Length 35. Print first 20):
x_0: 100.000000 100.000000 100.000000   0.000000   0.000000   0.000000
       0.000000 101.000000 101.000000 101.000000   0.000000   0.000000
       0.000000   0.000000 101.000000 101.000000 101.000000   0.000000
       0.000000   0.000000
Optimal vector x (Length 35. Print first 20):
x_k: 150.980869 111.640121  98.160034  39.340749  14.480087   0.000000
       2.839966  91.666799 109.637854 100.616747  21.369694  23.501194
       0.000000   3.223219  96.602949 105.792685 101.207685  12.179958
      28.086194   0.000000
Lagrange multipliers v. Vector length 80 (Print first 20):
v_k:   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   1.499744e+01
       0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
       1.500099e+01   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
       0.000000e+00   1.500334e+01
Gradient g_k Largest abs(gradient)          10.00000004324422065 (Print first 20):
g_k:   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00   2.000000e+00   5.000000e+00
       1.000000e+01   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00   2.000000e+00
       5.000000e+00   1.000000e+01   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00
       2.000000e+00   5.000000e+00
cErr  -1.390124e+00  -1.626082e+00  -9.397260e-01  -7.219411e-01  -8.205969e-01  -8.411106e-02
      -1.537422e-01  -2.609129e-01  -5.493371e-01  -1.486838e+00
```

```
10 inequalities off more than cTol =       1.000e-06
Worst constraint validation =       1.626e+00 for constraint # 2
Projected gradient gPr:  Largest abs(Projected gradient)           0.00554741660237834 (Print first 20):
gPr:   3.672439e-03   1.993014e-04   6.391490e-05   3.473138e-03   1.353865e-04   1.776357e-15
      -6.391490e-05   5.425947e-04   7.155117e-04   1.151163e-04   3.300221e-03   7.357819e-04
      -8.881784e-16  -1.790312e-04  -2.950168e-03   1.549876e-04   8.504858e-05   1.950649e-04
       8.057209e-04   5.329071e-15
*** WARNING: 30 reduced gradient values > 1E-5 *** Worst value: 5.547417e-03
    xVarLow 6  xVarLow 13 xVarLow 20 xVarLow 27 xVarLow 34 LinEq   1
    LinEq   2  LinEq   3  LinEq   4  LinEq   5  LinEq   6  LinEq   7
    LinEq   8  LinEq   9  LinEq   10 LinEq   11 LinEq   12 LinEq   13
    LinEq   14
v:   1.500006e+01   1.500018e+01   1.500026e+01   1.500077e+01   1.500291e+01  -4.963276e-01
    -1.830762e+01  -5.000418e+01  -4.994574e-01  -5.579803e+00  -4.000412e+01  -5.029502e-01
    -2.271442e+00  -3.000394e+01  -4.947600e-01  -1.396077e+00  -2.000368e+01  -4.975281e-01
    -1.213698e+00  -1.000291e+01

=== * * * ================================================= * * *


xk =

  150.9809
  111.6401
   98.1600
   39.3407
   14.4801
         0
    2.8400
   91.6668
  109.6379
  100.6167
   21.3697
   23.5012
         0
    3.2232
   96.6029
  105.7927
  101.2077
   12.1800
   28.0862
         0
    3.0155
   94.8390
   99.3846
  101.0268
    7.6343
   26.4440
         0
    3.9887
   85.2772
   88.0088
   98.7306
    4.9026
   15.7222
         0
    7.2581


fk =

   1.2719e+03
```

Problem is maybe infeasible

```
===== * * * ============================================================== * * *
TOMLAB /SOL - Time limited demonstration single user license          Valid to 2005-09-28
================================================================================
Problem: No Init File    -  1: User Problem 1   f_k     766.500000008777647054
                                         sum(|constr|)   1790.088087407047396482
                                f(x_k) + sum(|constr|)   2556.588087415825157223
                                              f(x_0)     766.500000000000000000

Solver: conSolve.  EXIT=1.  INFORM=103.
Schittkowski SQP algorithm with BFGS update
Close iterations, but constraints not fulfilled
Too large penalty weights to be able to continue
Problem is maybe infeasible

FuncEv  144 GradEv    4 ConstrEv  144 Iter    2
CPU time: 0.240000 sec. Elapsed time: 0.242363 sec.
Starting vector x (Length 35. Print first 20):
x_0: 101.000000 101.000000 101.000000   0.000000   0.000000   0.000000
       0.000000 102.000000 102.000000 102.000000   0.000000   0.000000
       0.000000   0.000000 102.000000 102.000000 102.000000   0.000000
       0.000000   0.000000
Optimal vector x (Length 35. Print first 20):
x_k: 101.000000 101.000000 101.000000   0.000000   0.000000   0.000000
       0.000000 102.000000 102.000000 102.000000   0.000000   0.000000
       0.000000   0.000000 102.000000 102.000000 102.000000   0.000000
       0.000000   0.000000
Lagrange multipliers v. Vector length 80 (Print first 20):
v_k:   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   3.920946e+01
       5.790539e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
       0.000000e+00   4.500000e+01   0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
       0.000000e+00   1.500020e+01
Gradient g_k Largest abs(gradient)          10.00000007020374149 (Print first 20):
g_k:   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00   2.000000e+00   5.000000e+00
       1.000000e+01   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00   2.000000e+00
       5.000000e+00   1.000000e+01   5.000000e-01   5.000000e-01   5.000000e-01   1.000000e+00
       2.000000e+00   5.000000e+00
cErr  -1.224730e+02  -1.146975e+02  -1.146975e+02  -1.067457e+02  -1.067457e+02   0.000000e+00
       0.000000e+00   0.000000e+00   0.000000e+00   0.000000e+00
5 inequalities off more than cTol =       1.000e-06
Worst constraint validation =       1.225e+02 for constraint # 1
Projected gradient gPr:  Largest abs(Projected gradient)           2.30666614030540362 (Print first 20):
gPr:   9.581610e-01   1.253162e+00   2.109424e-15  -2.950010e-01   1.253162e+00   2.664535e-15
       1.776357e-15   8.469864e-01   7.469444e-01   1.250027e+00  -1.949591e-01   7.500798e-01
       1.250027e+00   3.552714e-15   2.237829e-01   3.900431e-02  -1.250027e+00  -1.018047e-02
       2.039111e+00  -8.881784e-16
*** WARNING: 23 reduced gradient values > 1E-5 *** Worst value: 2.306666e+00
    xVarLow 6  xVarLow 7  xVarLow 14 xVarLow 20 xVarLow 21 xVarLow 27
    xVarLow 28 xVarLow 34 xVarLow 35 LinEq   1  LinEq   2  LinEq   3
    LinEq   4  LinEq   5  LinEq   6  LinEq   7  LinEq   8  LinEq   9
    LinEq   10
v:   5.503189e+00   9.496811e+00   1.374997e+01   6.289137e+00   8.710863e+00   5.160980e+00
     9.839021e+00   5.806666e+00   9.193334e+00   4.581610e-01   1.490002e+00   9.900021e-01
     3.469864e-01   7.431641e-01   1.493191e+00  -2.762171e-01  -5.067561e-01  -2.256783e+00
    -4.034515e-01  -4.676455e-01

=== * * * ============================================== * * *


xk =
```

```
  101.0000
  101.0000
  101.0000
    0.0000
    0.0000
         0
         0
  102.0000
  102.0000
  102.0000
    0.0000
    0.0000
    0.0000
         0
  102.0000
  102.0000
  102.0000
    0.0000
    0.0000
         0
         0
  103.0000
  103.0000
  103.0000
    0.0000
    0.0000
         0
         0
  103.0000
  103.0000
  103.0000
    0.0000
    0.0000
         0
         0

fk =

  766.5000
```