Iterative interaction between the schedule- and subsystem-level in semi-cyclic hybrid systems - A case study approach

A.J.M. van Heusden

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Iterative interaction between the schedule- and subsystem-level in semi-cyclic hybrid systems - A case study approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Committee: A.J.J. van den Boom, M. Kok

A.J.M. van Heusden

May 11, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

In this thesis, research is done on the influence and benefits of an iterative interaction between a scheduler and its subsystems for an updated scheduler which minimizes to a certain cost. This is done by providing a case study of a beer brewery. The scheduler is obtained by using a switching max-plus linear approach. The subsystems will be simulated, estimated and predicted using the system dynamics of the beer brewing case study. The processes discussed in the beer brewing process are mashing, brewing and fermentation. The simulation is done by filling in the system dynamics with addition of noise. The estimation is done by using an extended Kalman filter, and the predictions are done by filling in the system dynamics with the estimated states and no addition of noise. The updating of the scheduler is done by receiving the estimations and predictions of the subsystems and thereafter using model predictive control on the switching max-plus scheduler, also called model predictive scheduling. The results for the case study are shown as a substantiation of the conclusions drawn. Ultimately, discussions and further research are given for the reliability of the conclusions and extensions on the above summarized research.

Table of Contents

Chapter 1

Introduction

Modern day world often offers problems which can be modelled, analyzed and controlled as discrete event systems, referred to as DES's. In a significant amount of these DES's, conventional algebra can only formulate the problem as a non-linear one. A slight subclass of these non-linear systems can be reframed, under the condition that synchronization but no concurrency occurs, with the result of becoming linear in the max-plus algebra [1][8]. In existing literature a broad amount of applications can be found [9], such as max-plus models of ribosome dynamics during mRNA translation [2], max-plus modelling of manufacturing flow lines [12] and model predictive scheduling for container terminals [13].

Such max-plus linear systems are useful, since the analysis can be done by using techniques and properties quite familiar in conventional linear algebra. For instance the eigenvalues, eigenvectors, Cramer's Rule and spectral radius [1][8].

A relatively new subclass of these max-plus linear systems, are the switching max-plus linear systems[16]. This technique gives the opportunity to change between certain modes in a system, where for each mode a different state-space exists. This results in being able to change the structure of the system, and therefore adjust the synchronization or order of the system as will be elaborated later on in this report. Note that for this particular reason switching max-plus linear systems are a useful extension of max-plus linear systems. A quite remarkable coincidence is that in some cases, with the right conditions, similarities exist of these switching max-plus linear systems with well known hybrid systems [17]. Examples of applications of switching max-plus linear systems are changing recipes in production systems [14] and gait switching in legged locomotion [10].

(Switching) max-plus linear systems are extensively used for the purpose of scheduling. Normally, scheduling consists of making timetables for specific jobs or processes. These jobs or processes need to fulfill certain begin- and endtimes. For instance, in case of a railway network, the trains need to fulfill certain departure- and endtimes of the timetable it is made for. Most commonly, these schedulers are designed in a hierarchical way. This is illustrated in figure 1-1, where a scheduler sends certain information about the begin- and endtimes in

a hierarchical way from top to bottom.

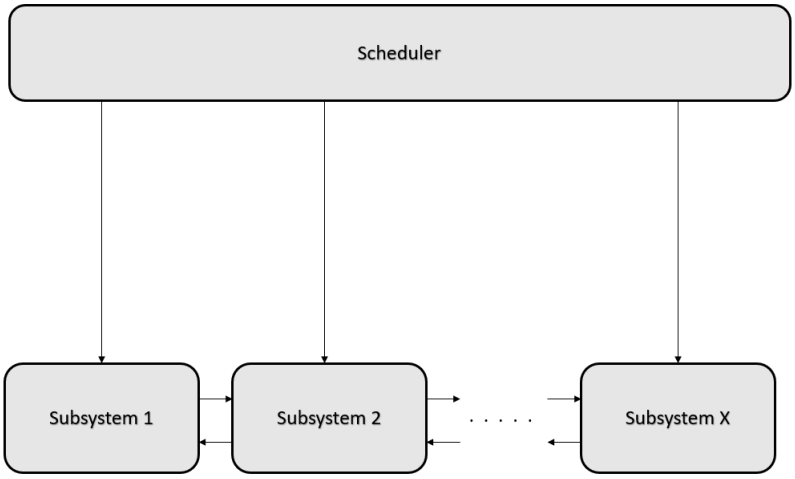

Figure 1-1: Hierarchical scheduler.

However, switching max-plus linear systems can adjust the timetable by assigning new routes, new orders, or adjust synchronizations for and between different trains. Continuously optimizing such scheduling problems with a high amount of for instance trains and different possibilities for the trains to come from place A to place B, can be computationally very hard though. For this purpose, iterative interaction within a certain time-step is proposed in [15]. In the referred study, the iterative interaction between the schedule- and subsystemlevel is described via the use of an interface. This interface receives information from both levels as is illustrated in figure 1-2. The question if this interface works, can already be agreed upon. Though, the exact processes and problems which are encountered within this interface, are worthy to look at.

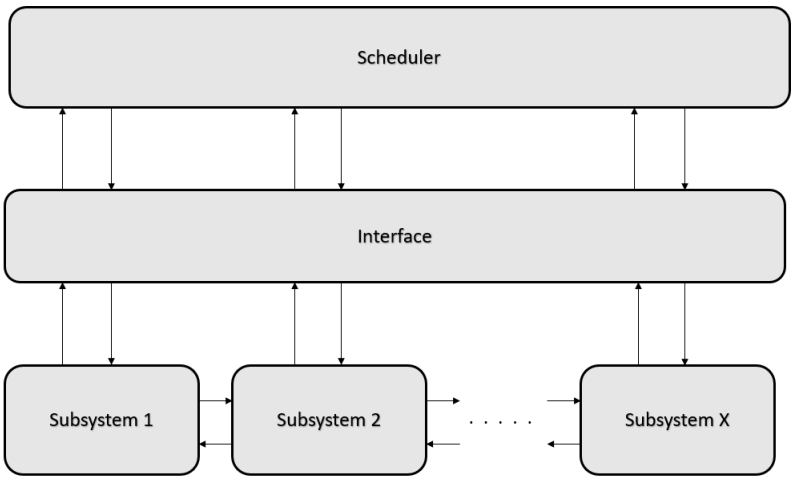

Figure 1-2: Iterative scheduler with adaptive behaviour.

The main scope of this thesis will therefore lie in the working of this iterative process. The interface is just an illustrative way to explain the problem, but essentially it can be translated to the research in the iterative manner the scheduler and subsystems communicate. Since a switching max-plus linear approach is used to model the scheduler, the top-level will work in the event-domain. The subsystem level however, will work in the (discrete)-time domain. The main problem will lie in the iterative communication between this event-domain top-level of the scheduler and the (discrete) time-domain bottom-level of the subsystems. This will be practically substantiated by the modelling of a case study. For the case study a beer brewery is proposed, where several processes are obtained as subsystems and the scheduler decides whether which subsystem should start. Underneath the main problem, several subproblems can be outlined as well.

The scheduler essentially consists of a switching max-plus framework, where different modes are assigned to different routes through the resources in which the processes take place in the case study. The modelling of these routes, as well as the choice for certain nominal process times in the switching max-plus matrices, must be thought of. Finally, these solution must be translated to initial begin- and endtimes for the subsystems.

When subsequently arriving at the subsystem-level which received these starting- and end-times, the subsystem need to start. Though, since real systems are not obtained, simulations need to be made with deviations in the nominal times for the scheduler to adapt to certain deviations. These deviations must not be too random, neither to close to the nominal process time. Considerations need to be made with respect to the process time.

Furthermore, when realizations are made for the simulations of the subsystem, an observer need to estimate the states of the subsystems. An approach is proposed which serves as an observer for the simulated subsystems. Afterwards, these estimated states, are used to put in a prediction model for the subsystems, which can be send thereafter to the scheduler again for the predicted process times.

Ultimately arriving again at the scheduler level, the predicted and updated process times are used to come up with an updated schedule, and so on. Actually, the subsystems can be looked upon as a big observer for the system matrices of the scheduler for the process times of the subsystems. The update of the scheduler is an important topic. This update can be considered using an event-based approach, or a time-based approach, as will be elaborated in this report.

To conclude, the outline of this report is given. First in chapter 2, the top-level of the schedule is discussed by giving background information with respect to the event-domain. Scheduling using a switching max-plus approach is considered as well. Afterwards, in chapter 3, the bottom-level is discussed by giving background infromation with respect to the time-domain. General approaches for the simulations, estimations and predictions of the subsystems are given. In chapter 4, the case-study is outlined to practically illustrate the working of the iterative process between the scheduler and the subsystems. In this chapter, mainly the methods and tools from chapter 2 and 3 are applied to the model of the case study. Thereafter in chapter 5, the results of these applied methods are given. Finally in chapter 6, conclusions and discussions are given, to summarize this thesis report, and give extensions

for further research.

Chapter 2

Background knowledge - Event domain for the scheduler

In systems which behave like discrete event systems, the system does not behave according to a certain time index t, but rather to an event- cycle- or batchcounter, denoted by the variable k. To remain consistent, in the whole report, k denotes the event-, cycle- or batchcounter, where for the discrete time-counter, the variable κ is chosen. First the background is given for the max-plus algebra, whereafter model predictive control for max-plus linear systems is obtianed for the update of the scheduler. Finally, the scheduler itself is outlined with a switching max-plus approach.

2-1 Max-plus algebra

2-1-1 Max-plus algebra

In this thesis, the scheduling process will be done by using max-plus algebra. Basically max-plus algebra consists of the operations maximization and addition, which replace the conventional operations addition and multiplication. Many of the scheduling problems can then be recasted linearly in the max-plus format. The following operations can be done in max-plus algebra [7].

$$\begin{aligned} a \oplus b &= \max(a, b) \\ a \otimes c &= a + c \end{aligned} \tag{2-1}$$

For $a, b, c \in \mathcal{R}_\varepsilon = \mathcal{R} \cup -\infty$, also called the max-plus semi-ring. Note that the inverse of the $\oplus$-operator does not exist, namely when having the outcome b when having the maximization of $\max(a, b)$, one can never find a. The only thing known is that $a \leq b$. The above equations can be extended to matrix algebra as:

$$\begin{aligned} (A \oplus B)_{ij} &= a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij}) \\ (A \otimes C)_{ij} &= \bigoplus_{k=1}^{n} a_{ik} \oplus c_{kj} = \max_{k=1,2,\ldots,n}(a_{ik} + b_{kj}) \end{aligned} \tag{2-2}$$

Furthermore the zero- and one-operator are replaced by respectively $\varepsilon = -\infty$ and $e = 0$. This makes sense as will be shown in the following example compared with conventional algebra:

$$\begin{aligned} a + 0 &= a \\ a \oplus \varepsilon &= \max(a, -\infty) = a \\ \\ a \cdot 0 &= 0 \\ a \otimes \varepsilon &= a + -\infty = -\infty = \varepsilon \end{aligned} \tag{2-3}$$

And for the one-operator the same procedure:

$$\begin{aligned} a \times e &= a \\ a \otimes e &= a + 0 = a \end{aligned} \tag{2-4}$$

And so on. The matrices belonging to respectively the zero- and identity-matrices, can also be obtained. For the zero-matrix it holds $\mathcal{E}_{ij} = \varepsilon$ everywhere and for the identity-matrix it holds $(E_n)_{ij} = \varepsilon$ and $(E_n)_{ii} = 0$ for $\forall i, j$. Finally the powers of a matrix are considered. In max-plus algebra the powers can be represented by the multiplication operator in conventional algebra:

$$A^{\otimes^k} = A \otimes A^{\otimes^{(k-1)}} = A \otimes A \otimes A^{\otimes^{(k-2)}} = k \cdot A \tag{2-5}$$

Where $A^{\otimes^0} = E_n$. Following up on the max-plus powers of a matrix, an important operator is introduced. The Kleene-star product namely, is useful in solving max-plus linear equation. The Kleene-star product is given by:

$$A^* = E_n \oplus A^{\otimes^1} \oplus A^{\otimes^2} \oplus A^{\otimes^3} \oplus .... \tag{2-6}$$

Ultimately, for the use of optimization in switching max-plus linear systems, max-plus binary variables are introduced with their adjoint as:

$$u = \begin{cases} \varepsilon \\ 0 \end{cases} \tag{2-7}$$

and

$$\bar{u} = \begin{cases} 0 \text{ when } u = \varepsilon \\ \varepsilon \text{ when } u = 0 \end{cases} \tag{2-8}$$

In conventional algebra, the max-plus binary variables can be translated as:

$$v_\varepsilon = \varepsilon(1 - v) \tag{2-9}$$

Where v is a normal binary variable, such that if $v = 0$, $v_\varepsilon = \varepsilon$ and if $v = 1$, v_ε turns out to be 0. Later on, we will see this is important to model the max-plus algebra into conventional solvers.

2-1-2 Max-plus linear systems

Max-plus algebra is mainly used in the sense of recasting non-linear problems when using conventional algebra into linear problems in max-plus algebra. Many techniques from linear

algebra can be translated to the max-plus algebra. As well state-space descriptions can be translated into max-plus algebra. This will result in:

$$\begin{aligned} x(k+1) &= A \otimes x(k) \oplus B \otimes u(k) \\ y(k) &= C \otimes x(k) \end{aligned} \tag{2-10}$$

In this framework, the variable k does not represent time-instances, but is represented as the batch- job- or cyclecounter. For instance, in a railway network it could represent the k-th train, or in an industrial plant, it could represent the k-th batch which need to be processes in the plant. In most of the linear systems in max-plus algebra, the states $x(k)$ represent time-instances. The A- and B-matrix normally represent processing times or travelling times. The input $u(k)$ represents the time-instant batch/product k comes in. This input can be controllable but does necessarily has to be, it could for instance also be considered as luggages coming at random time-instances into a bagage handling system. The output of the system $y(k)$ can be considered as the time-instant the batch k comes out of a system. In the rest of the report, to avoid misunderstandings with the states of the subsystems, the max-plus state $x(k)$ is replaced by $z(k)$ such that:

$$\begin{aligned} z(k+1) &= A \otimes z(k) \oplus B \otimes u(k) \\ y(k) &= C \otimes z(k) \end{aligned} \tag{2-11}$$

2-1-3 Switching max-plus linear systems

The extended class of max-plus linear systems, is the switching max-plus linear systems. In this case the matrices can change by assigning them to a certain mode. In every mode, a different A or B matrix exists. Furthermore the changing of the modes can be done by a switching rule. The state-space description will look like:

$$\begin{aligned} z(k+1) &= A^{\ell(k)} \otimes z(k) \oplus B^{\ell(k)} \otimes u(k) \\ y(k) &= C^{\ell(k)} \otimes z(k) \end{aligned} \tag{2-12}$$

Where the switching rule can be described for the mode $\ell(k)$ as:

$$\ell(k) = \phi(z(k-1), \ell(k-1), u(k), v(k)) \tag{2-13}$$

It could be dependant on the previous state, the previous mode, the input $u(k)$ and/or a control variable $v(k)$ which later on will be elaborated.

2-2 Model predictive control

The method for Model Predictive Control(MPC) is described in [6]. Essentially, it is a control technique in which a cost function is minimized over a certain horizon for a system with a specific state-space description. A big advantage of MPC, is the easy addition of several constraints to the control problem. MPC is widely described in common literature, since the

use can be extended to various types of different systems. The most simple system it can be used for are linear systems:

$$\begin{aligned} z(k+1) &= Az(k) + Bu(k) \\ y(k) &= Cz(k) \end{aligned} \tag{2-14}$$

A certain cost function is added, which can be minimized over certain tunable control variables. Usually the control variable is described as $u(k)$. Normally, the cost function is divided for a part which minimizes some cost w.r.t. the output of the system, and some cost which minimizes w.r.t. the input. This can be described as:

$$J(k) = J_{out}(k) + \lambda J_{in}(k) \tag{2-15}$$

Where λ is a scaling factor in which way the attention need to be lied upon respectively the input or output cost functions. A common output function is one with minimizing the difference between a certain reference signal which can be seen as a due date or departure time, where the input usually is minimized on its negative counterpart, which means the system need to operate with the input as late as possible. In equations this boils down to:

$$J(k) = \sum_{j=1}^{N_p} |(y(k+j|k) - r(k+j))| - \lambda \sum_{j=1}^{N_p-1} u(k+j) \tag{2-16}$$

The terms for $y(k+j|k)$ can be found by recursively fill in the state-space description:

$$\begin{aligned} y(k+1|k) &= Cz(k+1|k) = CAz(k|k) + CBu(k) \\ y(k+2|k) &= Cz(k+2|k) = CA^2z(k|k) + CBu(k+1) + CABu(k) \end{aligned} \tag{2-17}$$

Such that estimates can be put in vector form as:

$$\tilde{y}(k) = \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+N_p|k) \end{bmatrix} = H \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+Np-1) \end{bmatrix} + \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N_p} \end{bmatrix} z(k) \tag{2-18}$$

With H as:

$$H = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1} & CA^{N_p-2} & \dots & CB \end{bmatrix} \tag{2-19}$$

Now the cost function can be minimized with respect to the tunable control variables $u(k), ..., u(k+N_p-1)$. Addition of constraint can be done by assigning the linear constraint matrices:

$$F\tilde{u}(k) + G\tilde{y}(k) \leq h(k) \tag{2-20}$$

Often the control horizon is introduced. Since optimizing over a prediction horizon can be computationally hard and not adding a lot to the optimal solution, the control inputs are considered constant after the control horizon with $N_c \leq N_p$ such that:

$$u(k+j) = u(k+N_c-1) \text{ for } j = N_c, ..., N_p - 1 \tag{2-21}$$

Ultimately, minimizing the cost function gives a certain control input as optimal input for the problem. Filling in this control input and repeating all the steps at the next time-instances $k+1$ up till $k+N_p$ is basically what model predictive control does.

The above format can easily be extended to max-plus algebra. The state-space can be considered as:

$$\begin{aligned} z(k+1) &= A \otimes z(k) \oplus B \otimes u(k) \\ y(k) &= C \otimes z(k) \end{aligned} \tag{2-22}$$

Repeating the same steps as above, only now with the max-plus operators gives:

$$\tilde{y}(k) = \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+N_p|k) \end{bmatrix} = H \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+Np-1) \end{bmatrix} + \begin{bmatrix} C \\ C \otimes A \\ \vdots \\ C \otimes A^{\otimes N_p} \end{bmatrix} z(k) \tag{2-23}$$

With the matrix H as:

$$H = \begin{bmatrix} C \otimes B & 0 & \dots & 0 \\ C \otimes A \otimes B & C \otimes B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C \otimes A^{\otimes N_p - 1} & C \otimes A^{\otimes N_p - 2} & \dots & C \otimes B \end{bmatrix} \tag{2-24}$$

From [18], it can be concluded that this model predictive scheduling format, can be easily formulated in a Mixed-Integer Linear Problem(MILP) framework. This will be elabrated in chapter 4 of this report. It essentially comes down to rewriting all the $\oplus$-operators into seperate maximization constraints with linear terms in them, such that the overal problem will have a linear cost function, with linear constraints.

2-3 Scheduling with max-plus algebra

The following section summarizes the process of the scheduler framed into max-plus algebra. The scheduler is contained in the top level of the overall system and is considered to be in the event-based domain, denoted by event- cycle- or batchcounter k. How this is done, and which options there are regarding the scheduling process, is discussed in this section. Three options are proposed for the scheduling with switching max-plus linear systems. These are routing, ordering and at last synchronization. The 3 processes will be elaborated in this section with several examples, where afterwards the whole scheduling problem will be outlined. The following content is described in [18].

2-3-1 Routing

Routing is the process in which each job in a system needs to follow a specific route through the system. This can be compared for example with a railway network in which the intercity from Amsterdam to Eindhoven needs to follow another route than the intercity from Arnhem

to Groningen. Consider a system of M jobs, then for each job $j \in \{1, ..., M\}$, the job need to follow a specific set of operations where the operations p_j can be done on the resources $R_j = \{R_{j,1}, ..., R_{j,p_j}\}$ in processing order. The processing times are described by respectively $T_j(k) = \{\tau_{j,1}(k), ..., \tau_{j,p_j}(k)\}$. Now the state $z_j(k) = [z_{j,1}(k) \dots z_{j,p_j}(k)]^T$ can be considered as the starting times of each of the operations of job j. If we consider the processing times to be always greater than 0, the following inequalities can be obtained for job j:

$$z_{j,m}(k) \geq z_{j,l}(k) + \tau_{j,l} \tag{2-25}$$

With $l < m$ and $\tau_{j,l} > 0 \; \forall j, l$. Converting this to a matrix for whole job j, this results in:

$$\underbrace{\begin{bmatrix} z_{j,1}(k) \\ z_{j,2}(k) \\ \vdots \\ \\ z_{j,p_j}(k) \end{bmatrix}}_{z_j(k)} \geq \underbrace{\begin{bmatrix} \varepsilon & \varepsilon & \dots & & \varepsilon \\ \tau j, 1(k) & \varepsilon & \dots & & \varepsilon \\ \varepsilon & \tau_{j,2} & \varepsilon & \dots & \varepsilon \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \varepsilon & \dots & \varepsilon & \tau_{j,p_j-1} & \varepsilon \end{bmatrix}}_{A_0^{rout,j}} \otimes \underbrace{\begin{bmatrix} z_{j,1}(k) \\ z_{j,2}(k) \\ \vdots \\ \\ z_{j,p_j}(k) \end{bmatrix}}_{z_j(k)} \tag{2-26}$$

Or in a simple equation:

$$z_j(k) = A_0^{rout,j} z_j(k) \tag{2-27}$$

Now eventually, there will be a total of M jobs for which each a matrix can be obtained. The matrix for the routing of the whole set of jobs will look like:

$$\underbrace{\begin{bmatrix} z_1(k) \\ z_2(k) \\ \vdots \\ \\ z_M(k) \end{bmatrix}}_{z(k)} \geq \underbrace{\begin{bmatrix} A_0^{rout,1} & \mathcal{E} & \dots & & \mathcal{E} \\ \mathcal{E} & A_0^{rout,2} & \mathcal{E} & \dots & \mathcal{E} \\ \vdots & \ddots & \ddots & & \vdots \\ & & & & \mathcal{E} \\ \mathcal{E} & \dots & \mathcal{E} & & A_0^{rout,M} \end{bmatrix}}_{A_0^{rout}} \otimes \underbrace{\begin{bmatrix} z_1(k) \\ z_2(k) \\ \vdots \\ \\ z_M(k) \end{bmatrix}}_{z(k)} \tag{2-28}$$

Note that these equations are all dependant on the cycle k which is not necessarily the case. In much practical situations, routes from jobs can also be travelled via previous cycles. In the case routing is dependant on the previous cycle, the routing matrix equation will become:

$$z(k) \geq A_0^{rout} \otimes z(k) \oplus A_1^{rout} \otimes z(k-1) \tag{2-29}$$

In a production system this is often not necessary since there is only one job per cycle, as k represents the product/batch counter. Though, alternate routes for the specific job can be thought of as well. For instance when making a product in a production system, multiple machines can be used. Or in a railway network, where for each individual train a routing matrix is build, trains could have different routes from Arnhem to Groningen or Amsterdam to Eindhoven. Therefore the routing matrices can be different for different routes. In the general case there will be L alternative routes, then for each of the routes, we can obtain the matrices $A_{\mu,l}$ with $\mu \in \{0, 1\}$ and $l \in \{1, ..., n_l\}$ where n_l is the total number of alternative routes.

2-3-2 Ordering

Ordering is the process in scheduling where on the same resource, operations can be ordered respectively after or before each other. This is different from the routing, where in routing the route is defined for specific routes on different resources, whereas ordering is the process in which different operations from different jobs are ordered in a specific sequence. As an example, this can be seen as 2 trains following the same route from Rotterdam to Dordrecht for instance, but where the order of the 2 trains is changed via a station in between. This could happen when a faster train is behind a slower train.

In the ordering process, n operations are considered divided over N resources. When still considering L alternative routes, parametrizations can be done by assigning max-plus binary variables $w(k)$ for each of the routes. Let $P_l \in \mathcal{B}_\varepsilon^{n\times n}$ with $\mathrm{l} \in \{1, ..., L\}$ be a mtrix with max-plus binary entries where $[P_l]_{i,j} = 0$ if operation i and operation j are executed on the same resource and $[P_l]_{i,j} = \varepsilon$ if operation i and j are executed on different resources. Then the following matrix can be obtained:

$$P(w(k)) = \bigoplus_{l=1}^{L} w(k) \otimes P_l \tag{2-30}$$

Now for the following part, let $H(k)$ be considered as the seperation matrix, where $H_{i,j}(k) \neq \varepsilon$ is the seperation time between operation i and j if they may be scheduled on the same resource and $H_{i,j}(k) = \varepsilon$ if operation i and j can never be scheduled on the same resource. Ultimately the matrices $\Gamma_\mu(k)$ are introduced with $\mu = 0, 1$. These will be order decision matrices with max-plus binary variables. $[\Gamma_\mu(k)]_{i,j} = 0$ if operation i in cycle k is scheduled after operation j in cycle $k + \mu$ and $[\Gamma_\mu(k)]_{i,j} = \varepsilon$ if operation i in cycle k is scheduled before operation j in cycle $k+\mu$. By defining $\gamma_\mu(k)$ as the vector with the stacked column vectors of matrix $\Gamma+\mu(k)$ such that $\gamma_\mu(k) = vec(\Gamma_\mu(k))$, we can use this to obtain the notation $\Gamma_\mu(k) = \Gamma(\gamma_\mu(k))$. Now adding up all the ordering matrices, one can come up with the following matrices:

$$A_\mu^{ord}(w(k), \gamma_\mu(k)) = P(w(k)) \odot \Gamma(\gamma_\mu(k)) \odot H(k) \tag{2-31}$$

Ultimately the operation ordering constraints in the system can be formulated by maximization such that:

$$z(k) \geq A_0^{ord}(w(k), \gamma_0(k)) \otimes z(k) \oplus A_1^{ord}(w(k), \gamma_1(k)) \otimes z(k-1) \tag{2-32}$$

2-3-3 Synchronization

Synchronization is the process in which different jobs need to be synchronized with each other for some kind of reason whatsoever. For instance, in a railway network, this can be seen as transfers from one train to the other, where these transfers must allow a certain time for the passengers to get from one train to the other. The train themselves can be observed as jobs, but a certain synchronization between those trains must occur for people to transfer. Whereafter the trains can continue doing their job, i.e. following their specific route. In mathematical representation this can be seen set as the following rule. First we can define a

number of modes for the synchronization $l = 1, ..., L_{synch}$, where for every mode we obtain a system matrix for $\mu = 0, 1$:

$$[A^{synch}_{\mu,l}(k)]_{i,j} = \begin{cases} 0 & \text{if operation } i \text{ in cycle } k \text{ is to be scheduled behind operation i in cycle k-}\mu \\ \varepsilon & \text{elsewhere} \end{cases} \tag{2-33}$$

Now the synchroiuzation constraints with respect to the different operations in the system can be formulated as:

$$z(k) \geq A^{synch}_0(s_0(k)) \otimes z(k) \oplus A^{synch}_1(s_1(k) \otimes z(k-1) \tag{2-34}$$

Where for $\mu = 0, 1$ the following can be given:

$$A^{synch}_\mu(s_\mu(k)) = \bigoplus_{l=0}^{L_{synch}} [s_\mu(k)]_l \otimes A^{synch}_{\mu,l}(k) \tag{2-35}$$

$s_\mu(k)$ are max-plus binary variables for scheduling the synchronization. $[s_\mu(k)]_l = 0$ has the meaning that synchronization l is made and $[s_\mu(k)]_l = \varepsilon$ has the meaning synchronization l is cancelled.

2-3-4 Overal scheduler

The overal scheduler has the decision variables $w(k)$, $\gamma_\mu(k)$ and $s_\mu(k)$ for $\mu(k) = 0, 1$. Stacking these into one vectorwill give:

$$v(k) = \begin{bmatrix} w(k) \\ \gamma_{\mu 0}(k) \\ \gamma_{\mu 1}(k) \\ s_{\mu 0}(k) \\ s_{\mu 1}(k) \end{bmatrix} \in \mathcal{B}^{L_{tot}}_\varepsilon \tag{2-36}$$

Where L_{tot} represents the total number of scheduling variables. Now we can formulate all the above information from the previous paragraphs into one equation:

$$x(k) \geq A_0(v(k)) \otimes x(k) \oplus A_1(v(k)) \otimes x(k-1) \tag{2-37}$$

Where the matrices $A_\mu(v(k))$ are made by combination of the following matrices:

$$\begin{aligned} A_\mu(v(k)) &= A^{job}_\mu(w(k)) \oplus A^{ord}_\mu(w(k), \gamma_\mu(k)) \oplus A^{synch}_\mu(s_\mu(k)) \\ &= \textstyle\bigoplus_{l=1}^{L_{tot}} v_l(k) \otimes A_{\mu,l}(k) \end{aligned} \tag{2-38}$$

Now by introducing inputs and reference signals. Inputs can be often seen as starting times of jobs, and reference signals as lower bounds of certain operations to start. This can be quite easily be added to the above equation:

$$\begin{aligned} z(k) &\geq B(v(k)) \otimes u(k) \\ z(k) &\geq r(k) \end{aligned} \tag{2-39}$$

Such that the overal equation with inputs and reference signals included can be formulated as:

$$z(k) \geq A_0(v(k)) \otimes z(k) \oplus A_1(v(k)) \otimes z(k-1) \oplus B(v(k)) \otimes u(k) \oplus r(k) \tag{2-40}$$

2-3-5 Optimization of the case study

The optimization in a scheduler lies in the fact a cost function need to be minimized with the above maximization can be put in the optimization as constraint matrices. The outcome of the optimization will result in the perfect decision variables $w(k)$ and $\gamma_\mu(k)$ for each cycle k. In the case study, these binary variables are formulated as $v(k)$ and $s(k)$. Since scheduling problems can become quite complex with respect to the number of decision variable, the case study in this thesis is provided with a problem only regarding the routing and ordering in a specific scheduling problem. Such that the constraint matrices can be reformed into:

$$z(k) \geq A_0(w(k)) \otimes z(k) \oplus A_1(w(k)) \otimes z(k-1) \otimes B(w(k)) \otimes u(k) \\ \bigoplus_{\mu=\mu_{min}}^{\mu_{max}} A_{ord,\mu}(w(k), \gamma_\mu(k)) \otimes z(k-\mu) \tag{2-41}$$

In the dynamical scheduler, the above equation will be a function of inputs given by the estimated arrival times of the subsystems. This will be practically explained in the case study paragraphs.

Chapter 3

Background knowledge - Time domain for the subsystems

In this chapter, the main tools and background knowledge is discussed for the understanding and modelling of the subsystems. The subsystems are modelled in the time-domain. To avoid misunderstanding, the state will be modelled as $x(\kappa)$ where the state for the event-based domain is modelled as $z(k)$. As is already mentioned, κ stands as the discrete time-counter, where k stands for the event-counter. First the Euler-method is given, such that not too complex non-linear continuous time systems can be discretized. Afterwards the Extended Kalman Filter(EKF) is given as the observer for the subsystem such that the states can be estimated accordingly. Finally the simulations, estimations and predictions for the subsystems are discussed with some illustrative figures. The elaboration in the next chapter for the case study will stick more or less to the same manner namely.

3-1 Euler Method

Essentially the Euler method is the easiest way to come up with a numerical solution to a differential problem. This method basically consists of the first order Taylor expansion of a function. First a simple differential equation is considered:

$$\begin{aligned} \tfrac{dy(t)}{dt} &= f(t, y(t)) \\ y(0) &= y_0 \end{aligned} \tag{3-1}$$

The Taylor expansion is given by:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2}h^2y''(t) + \frac{1}{3!}h^3y'''(t) + ... + \frac{1}{n!}h^ny^n(t) \tag{3-2}$$

Where the higher order terms will be less and less involved if the time-step h is taking smaller to 0. For the Euler method. Only the first 2 terms are used:

$$y(t+h) = y(t) + hy'(t) \tag{3-3}$$

Now this can be reframed by using equation 3-1:

$$y(t+h) = y(t) + hf(t, y(t)) \tag{3-4}$$

Now by setting up the algorithm for finding each next value for y_t the numerical solution can be obtained starting with the initial value y_0 untill the end time of the time interval in which one wants to find the numerical solution.

$$\begin{aligned} y(\kappa+1) &= y(\kappa) + hf(\kappa, y(\kappa)) \\ y(0) &= y_0 \\ t(\kappa+1) &= t(\kappa) + h \end{aligned} \tag{3-5}$$

3-2 Extended Kalman Filtering

The method for extended Kalman filtering is taken from [4]. Considering a non-linear system in the form of:

$$\begin{aligned} x(\kappa+1) &= f(\kappa, x(\kappa)) + H(\kappa, x(\kappa))w(\kappa) \\ v(\kappa) &= g(\kappa, x_\kappa) + \eta(\kappa) \end{aligned} \tag{3-6}$$

Where $f(\kappa)$ and $g(\kappa)$ are vector-valued functions with ranges $\mathcal{R}^n$ and $\mathcal{R}^q$, with values $1 \leq q \leq n$ and $H(\kappa, x(\kappa))$ a matrix-valued function with range in $\mathcal{R}^n \times \mathcal{R}^q$, such that for each κ the first order partial derivatives of $f(\kappa, x_\kappa)$ and $g(\kappa, x_\kappa)$ with respect to all the components of $x(\kappa)$ are continuous. For the noise sequences of $w(\kappa)$ and $\eta(\kappa)$, we consider zero-mean white noise processes such that:

$$\begin{aligned} &E(w(\kappa)w(l)^T) = Q(\kappa)\delta(\kappa, l), \quad E(\eta(\kappa)\eta(l)^T) = R(\kappa)\delta(\kappa, l) \\ &E(w(\kappa)\eta(l)^T) = 0, \qquad\qquad E(w(\kappa)x(0)^T) = 0, \\ &E(\eta(\kappa)x(0)^T) = 0 \end{aligned} \tag{3-7}$$

For all κ and l. The terms $w(\kappa) \sim (0, Q(\kappa))$ and $\eta(\kappa) \sim (0, R(\kappa))$ stand for the white noise signals $w(\kappa)$ and $\eta(\kappa)$ having a mean of 0, and covariance matrix of respectively $Q(\kappa)$ and $R(\kappa)$. By initializing for the estimates, the following is given:

$$\hat{x}(0) = E(x(0)), \quad \hat{x}(1|0) = f(0, \hat{x}(0)) \tag{3-8}$$

Now the following algorithm can be obtained for estimating all the states using the measurements $v(\kappa)$ and information about the system dynamics as described above. First initialization is done by:

$$P(0|0) = \mathrm{Var}(x(0)), \quad \hat{x}(0) = E(x(0)) \tag{3-9}$$

Now by increasing the step κ all the way to the maximum κ in which measurements are obtained, the following algorithm can be obtained to give a minimum-variance unbiased estimate

of the state up till κ.

$$\begin{aligned} P(\kappa|\kappa-1) = [\tfrac{\partial f(\kappa-1)}{\partial x(\kappa-1)}(\hat{x}(\kappa-1))]P(\kappa-1|\kappa-1)[\tfrac{\partial f(\kappa-1)}{\partial x(\kappa-1)}(\hat{x}(\kappa-1))]^T \\ +H(\kappa-1,\hat{x}(\kappa-1))Q(\kappa-1)H(\kappa-1,\hat{x}(\kappa-1))^T \end{aligned}$$

$$\hat{x}(\kappa|\kappa-1) = f(\kappa-1,\hat{x}(\kappa-1))$$

$$G(\kappa) = P(\kappa|\kappa-1)[\tfrac{\partial g(\kappa)}{\partial x(\kappa)}(\hat{x}(\kappa|\kappa-1))]^T\Big[\tfrac{\partial g(\kappa)}{\partial x(\kappa)}(\hat{x}(\kappa|\kappa-1))]P(\kappa|\kappa-1)[\tfrac{\partial g(\kappa)}{\partial x(\kappa)}(\hat{x}(\kappa|\kappa-1))]^T + R(\kappa)\Big]^{-1}$$

$$P(\kappa|\kappa) = \Big[I - G(\kappa)[\tfrac{\partial g(\kappa)}{\partial x(\kappa)}(\hat{x}(\kappa|\kappa-1))]\Big]P(\kappa|\kappa-1)$$

$$\hat{x}(\kappa|\kappa) = \hat{x}(\kappa|\kappa-1) + G(\kappa)(v(\kappa) - g(\kappa,\hat{x}(\kappa|\kappa-1))) \tag{3-10}$$

The whole derivations of the extended Kalman filter can be found in the prescribed citation [4].

3-3 Modelling of the subsystems

In the following sections, the main idea behind the simulations, estimations of the simulations, and the subsequent predictions is given. This is more to illustrate, whereafter in chapter 4, the state-spaces of the obtained processes are elaborated.

3-3-1 Data gathering and estimations

First of all, in this thesis, no real life systems are obtained and therefore no real life data can be measured. This means certain realizations need to be made regarding the processes of the subsystems. In general a subsystem can be described by a non-linear state-space description as:

$$\begin{aligned} \dot{x}(t) &= f(x(t),u(t)) \\ y(\kappa) &= h(x(\kappa)) \end{aligned} \tag{3-11}$$

Where the dynamics are assumed to be time-invariant, and the output is assumed to have no feedthrough term. The noise is considered to be additive. Further research could also focus on integrated noise, but this will not be discussed in this thesis. Furthermore, the noise is assumed to be zero-mean white noise. The state-space description with additive noise will look like:

$$\begin{aligned} \dot{x}(t) &= f(x(t),u(t)) + w(t) \\ y(\kappa) &= h(x(\kappa)) + v(\kappa) \end{aligned} \tag{3-12}$$

With mean and covariances as $w(t) \sim (0, Q)$ and $v(\kappa) \sim (0, R)$. The output measurements $y(\kappa)$ are considered to be discrete, since taking measurements from real systems are in general discrete. The simulation in computer programmes, can not be done by continuous state-space models. Therefore a discretization is made, with a small enough step size, such that the original continuous model behaves the same as the discrete one. Note that the choice for which method to use for the discretization of the continuous function, depends on whether the function is highly non-linear or not. In this thesis, fortunately, the functions are not highly non-linear and an easy discretization method with a not too small step size will fulfil.

The Euler-discretization method is used, which can be seen as the first order Taylor expansion as is described in the prelimaries. This will look like:

$$\begin{aligned} x(\kappa+1) &= x(\kappa) + T_s f(x(\kappa), u(\kappa)) + w(\kappa) \\ y(\kappa) &= h(x(\kappa)) + v(\kappa) \end{aligned} \tag{3-13}$$

Note that the covariance of the discrete zero-mean white noise signal $w(\kappa)$ changes with the covariance of the continuous time zero-mean white noise with a factor T_s^2. By making a noise sequence in MATLAB, one can come up with a realization of the prescribed system dynamics just by simply making a for-loop and iteratively updating the simulation from $x(\kappa)$ up till $x(\kappa+1)$ as can be observed in Appendix B-2-2 and B-2-7.

The subsystems contained in this thesis, are mainly exponential growth function, where a certain parameter p is subjected to some uncertainties. To explain the method for the estimations, first a very simple 2D-model is proposed[4], for which a certain parameter need to be estimated. Afterwards, in the chapters about the case study, this method is translated to the subsystems from the case study. The subsystems are elaborated with their own state-spaces in chapter 4. For now, a simple 2D-model is constructed for the explanation of the method. The following state space description is used, and the problem becomes to estimate all the states:

$$\begin{aligned} \begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+1) \end{bmatrix} &= \begin{bmatrix} x_2(\kappa)x_1(\kappa) \\ \zeta x_2(\kappa) \end{bmatrix} + \begin{bmatrix} 0 \\ w(\kappa) \end{bmatrix} \\ y(\kappa) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \end{bmatrix} + v(\kappa) \end{aligned} \tag{3-14}$$

To explain this in words, the estimation problem, is to estimate $x_1(\kappa)$ and $x_2(\kappa)$ which is subjected to process noise, by only receiving measurements from $x_1(\kappa)$ which are subjected to measurement noise. To do this, an extended Kalman filter is proposed, where in the previous section this method is already outlined. The estimator will not be optimal, since the non-linear model need to be linearized every iteration for the Kalman filter to work. Though as we will see later in the case study, it works sufficiently well. In the state-space description above ζ is considered as a memory factor, which value is $0 < \zeta < 1$. The choice for the values of the covariance matrices Q and R, as well as the choice for the value of the memory factor ζ are substantiated in chapter 5.

3-3-2 Predictions

Essentially the predictions are quite straightforward. By estimating the system its states at discrete time-instant κ for job/batch or cycle k such that the state variables $x_1(\kappa, k)$ and $x_2(\kappa, k)$, the states with respect to its own discrete time κ but also with respect to a specific event/cycle/batch k, are within a certain accuracy known, the system dynamics can be filled in with the expectation of the noise sequences. Since these expectations of the noise sequences are 0, predictions can be made by just filling in the states iteratively for the next time steps in the system dynamics without noise, and continue this step up till the end time of the process or a desired threshold.

The problem with making predictions for the subsystems, is accuracy. The scheduler can not do anything with predictions which are later on evidently so much off with respect to the real end time of the process. Clearly, a logical update of the scheduler can not be obtained with too much "randomness". To explain this, some simulations are shown for one of the subsystems. The predictions are made after time $\kappa = 5$ days but since the x-axis is in "milliDay", this means $\kappa = 5000$ mDay. First a low covariance of the process noise is chosen, where later on a 10 times higher covariance is chosen. Of course the seed for the random processes is taken the same, such that comparison can be done. This is just an illustrative example, later on in this report the results will be explained and further elaborated.

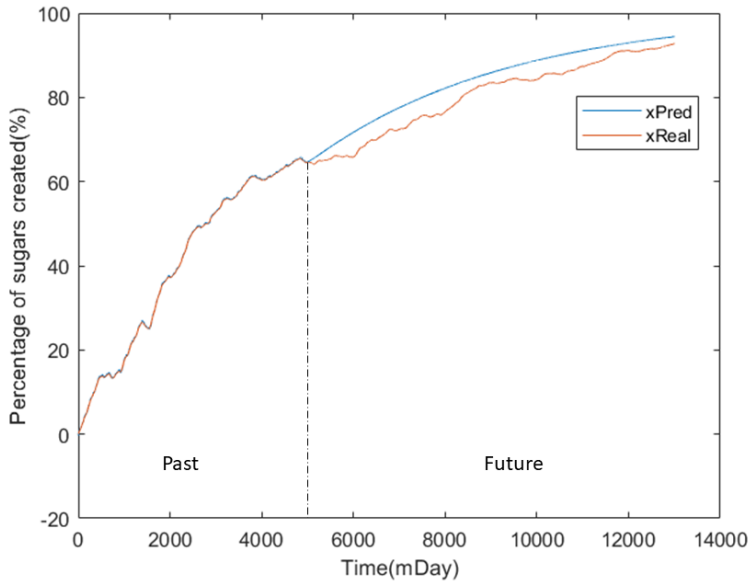

Figure 3-1: Low covariance of the process noise.

As can be seen in figure 3-1, the estimation of the system its states are quite nice up till $\kappa = 5000$ mDays. The prediction at time $\kappa = 5000$ mDays, the blue line, gives also quite a sufficient prediction with respect to the real data, the red line. Of course, one could argue if the prediction is only near the red line, because the prediction is made at quite a late stage. As will be elaborated in the results of the case study, the reliability of the prediction with

respect to the time it is been made on is also an aspect one has to look at. Now if we amplify the process noise its covariance with a factor 10, the figure 3-2 is obtained.

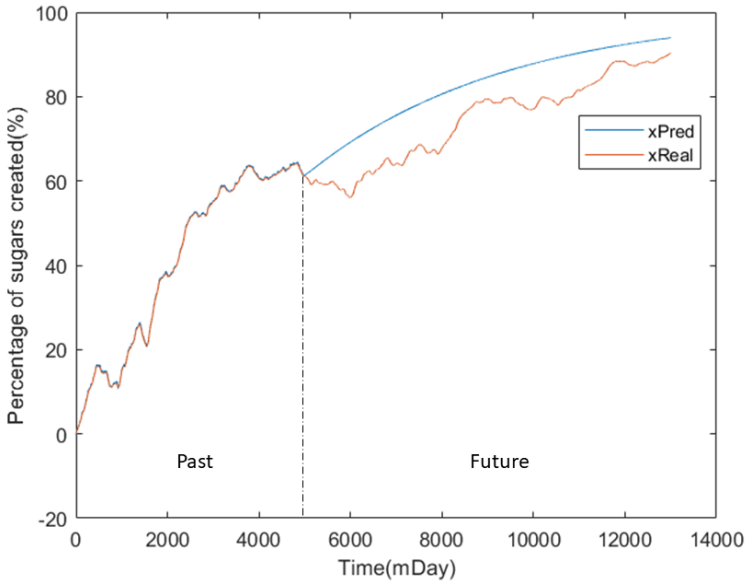

Figure 3-2: High covariance of the process noise.

Still the estimations up till time $\kappa = 5000$ mDays are very nice. Though the prediction after $\kappa = 5000$ mDays, is already having a much larger off-set from the real obtained trajectory. Of course this is also dependant on the realization of the sequence of the process noise, but the thing one would observe is that the off-set will become larger if the covariance of the noise increases. Of course this is logical, but it makes the prediction of the model rather useless with a too high covariance.

Besides the noise of the covariance matrix on an additional state which describes the behaviour of certain parameter, the "memory" factor ζ also plays a role in making the predictions. In the following graphs, the covariance is hold constant, but the ζ is changed from a factor just beneath 1, to a factor closer to 0 with respect to the value just beneath 1. In the first case the memory factor is taken as $\zeta = 0.99$, the results can be observed in 3-3. The model is made quite noisy such that the influence of the memory factor can be clearly observed. The estimations of the model are still very nice up till $\kappa = 5000$ mDays. The prediction could be better, but is still not useless. For the next figure, the memory factor is descreased to $\zeta = 0.95$, with the same process noise, i.e., the seed and covariance are the same for both cases. The results are shown in 3-4. What can be observed is that the same noise applied to the case $\zeta = 0.99$ is blushed out much faster. The overall model therefore becomes significantly less noisy, and as a result much better predictable. The problem which now occurs, is that the model becomes too little influenced by the noise. Since the model its data is simulated with these parameters and covariances as well, this will result in the processes not deviating a lot in the process end times. Since the goal of this thesis is to update a schedule due to processes of subsystems

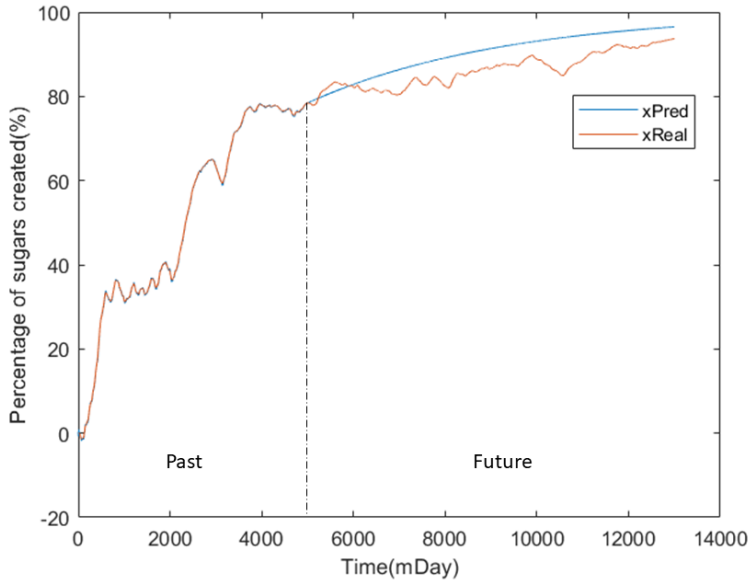

Figure 3-3: The memory factor taken $\zeta = 0.99$.

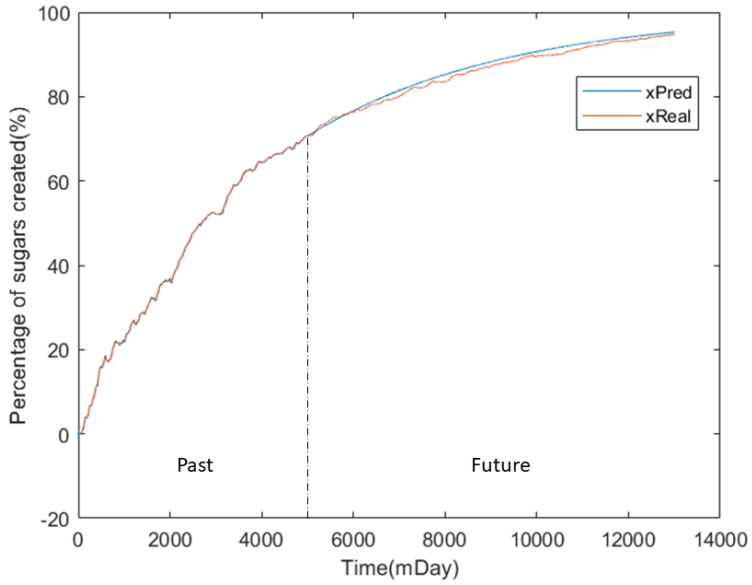

Figure 3-4: The memory factor taken $\zeta = 0.95$.

deviating, this will give the opposite effect. To conclude, a trade-off has to be made; The noise must not be too high for the system to be predictable, but the memory factor must not be too close to 0, since the model as a result does not deviate enough to have signicant differences in the process end times. Of course, a combination of the 2 could also work out,

increase the noise but let the memory factor also be close to 0. The trade-off between this 2 factors, will be elaborated in the case study chapter of this report. The trade-off is rather a tunable trial-and-error occasion, than a mathematically derived process.

Chapter 4

Case study - Methods

In this chapter, the case study is outlined. The case study, a beer brewery, is provided with several subsystems. The subsystems are elaborated, as well as the scheduler for the begin- and endtimes of these subsystems. This chapter is mainly used for the elaborations of the methods described in previous chapters as an application on the case study. The results are discussed in the next chapter.

4-1 Scheduling

4-1-1 The basic model

In this scheduling problem only routing and ordering in the scheduling problem is considered. Synchronization between different batches are not of any importance, and therefore not taken into account. Each of the processes has process time p_i with $i \in \{1, ..., 5\}$ since there are 5 machines. There are 2 machines for mashing, M_1 and M_2. 2 machines for brewing, M_3 and M_4, and 1 machine for the fermentation process, M_5. Since there are multiple machines for the same processes regarding the mashing and brewing process, 4 possible routes through the machines can be obtained:

- $M_1 - M_3 - M_5$ for $\ell = 1$
- $M_1 - M_4 - M_5$ for $\ell = 2$
- $M_2 - M_3 - M_5$ for $\ell = 3$
- $M_2 - M_4 - M_5$ for $\ell = 4$

In figure 4-1 the beer brewing factory is schematically illustrated, where the arrows represent each of the routes which can be followed. As can be seen, each of the routes must go through machine M_5. Therefore, the ordering process is considered in machine M_5 such that if in

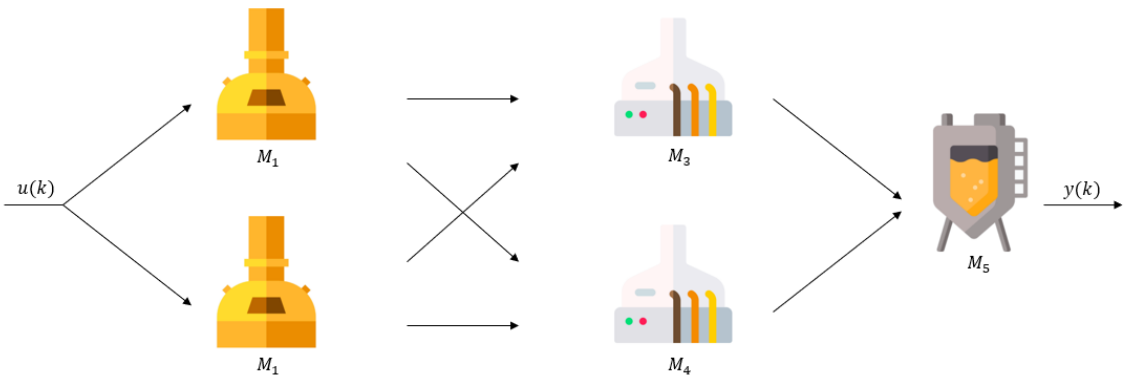

Figure 4-1: Schematic diagram of the beer brewing processes.

cycle k the job is very much delayed, the job in cycle $k+1$ can order itself before the job from cycle k in machine M_5. As can be observed in the model hereafter, the transportation time between the process tanks is neglected, where the transportation time would be insignificant compared with the process times of the machines. The machines will start working as soon as possible. The states $z_i(k)$ are the starting times of each of the machines $i \in \{1, ..., 5\}$. $u(k)$ represents the time instants a new batch comes in. For the first route the derivations are made, where for the other routes these derivations follow quite straightforward. The first route $M_1 - M_3 - M_5$ will look as follows. For machine M_1 to start, the batch $u(k)$ must be available and the previous batch need to be finished already, i.e. the previous batch its starting time with addition of the process time. Translated to equations this gives:

$$z_1(k) = \max(u(k), z_1(k-1) + p_1) \tag{4-1}$$

Or equivalently;

$$z_1(k) = z_1(k-1) \otimes p_1 \oplus u(k) \tag{4-2}$$

For the next machine M_3 in this specific route, the equations can easily be obtained. It can only start when the process in the previous machine is done, or when the process of the previous batch in the same machine is done. Mathematically:

$$z_3(k) = \max(z_1(k) + p_1, z_3(k-1) + p_3) \tag{4-3}$$

Or equivalenty:

$$z_3(k) = z_3(k-1) \otimes p_3 \oplus z_1(k) \otimes p_1 \tag{4-4}$$

And at last for machine M_5 the same procedure is done. However, since in machine 5 ordering can occur, such that jobs from different cycles can pass each other the constraint $z_5(k) \geq z_5(k-1) + p_5$ does not hold anymore. Later on, the derivations for the ordering process are derived. This derivations are just to illustrate how the model is obtained. For now, the constraint $z_5(k) \geq z_5(k-1) + p_5$ is left out of the constraint equations.

$$z_5(k) = z_3(k) + p_3 \tag{4-5}$$

Or equivalenty:

$$z_5(k) = z_3(k) \otimes p_3 \tag{4-6}$$

For the other machines, they will not start but be ready at all times since no job is done on these machines. Therefore a 0 is assigned in the max-plus matrices for the previous cycle $k-1$. In the previous cycle, for the cosntraints with respect to the state $z_5(k)$, an ε is assigned. The matrix with constraints for the previous cycle is mathematically formulated as A_1. The matrix for all maximizations with respect to the current/same cycle is formulated as A_0. At last, the matrix for the maximizations with respect to the input is formulated as B. The above equations can be written in matrix format for mode 1, the route $M_1 \rightarrow M_3 \rightarrow M_5$ is assigned to mode 1, as:

$$z(k) = \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} = \underbrace{\begin{bmatrix} p_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & p_3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}}_{A_1(\ell=1)} \otimes \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \\ z_3(k-1) \\ z_4(k-1) \\ z_5(k-1) \end{bmatrix} \oplus \underbrace{\begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ p_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & p_3 & \varepsilon & \varepsilon \end{bmatrix}}_{A_0(\ell=1)} \otimes \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} \oplus \underbrace{\begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}}_{B(\ell=1)} \otimes u(k) \tag{4-7}$$

Note that in this case, also the matrix A_1 is chosen to be in mode 1. This is not necessarily the case. since in the previous batch $k-1$ another mode could be active. Therefore, observe that mode sequences are necessary to obtain all the matrices. The matrices for the 3 other modes will be:

$$A_1(\ell=2) = \begin{bmatrix} p_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & p_4 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, A_0(\ell=2) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ p_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & p_4 & \varepsilon \end{bmatrix}, B(\ell=2) = \begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}$$

$$A_1(\ell=3) = \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & p_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & p_3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, A_0(\ell=3) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & p_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & p_3 & \varepsilon & \varepsilon \end{bmatrix}, B(\ell=3) = \begin{bmatrix} \varepsilon \\ 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix} \tag{4-8}$$

$$A_1(\ell=4) = \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & p_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & p_4 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, A_0(\ell=4) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & p_2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & p_4 & \varepsilon \end{bmatrix}, B(\ell=4) = \begin{bmatrix} \varepsilon \\ 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}$$

By using max-plus binary variables, all the modes can be assigned to a max-plus binary variable for each individual cycle k being $v_i(k) = 0$ or $v_i(k) = \infty$ for $i \in \{1, ..., 4\}$, since there are 4 possible routes. Ultimately the system its routing process evolves according to greater or equal constraints by making the equality signs with respect to maximization equal to greater or equal constraints:

$$z(k) \geq A_1(v(k-1)) \otimes z(k-1) \oplus A_0(v(k)) \otimes z(k) \oplus B(v(k)) \otimes u(k) \tag{4-9}$$

Note that the matrix $A(v(k-1))$ is not dependant on $v_i(k)$ but on $v_i(k-1)$ since this is done in the previous cycle in which a mode is chosen by the binary variables. The max-plus binary variables are assigned such that if one of the binary variables is equal to 0, the others will be equal to ε such that only one route per cycle can be chosen. Translating the different possible routes $v(k)$ to the binary variables is done by:

$$\begin{aligned} A_1(v(k-1)) &= \bigoplus_{\ell=1}^{4} v_\ell(k-1) \otimes A_1(\ell(k-1)) \\ A_0(v(k)) &= \bigoplus_{\ell=1}^{4} v_\ell(k) \otimes A_0(\ell(k)) \\ B(v(k)) &= \bigoplus_{\ell=1}^{4} v_\ell(k) \otimes B(\ell(k)) \end{aligned} \tag{4-10}$$

The above equations are only focussed on the routing of the scheduler. The ordering of the scheduler is done also by introducing max-plus binary variables. The starting time for machine M_5 is constrained in the routing process by being greater or equal to the endtime of the previous process for the same job in cycle k by $z_5(k) \geq z_3(k) + p_3$. Now new constraints for the ordering are added, with respect to all the jobs from cycles $k, ..., k+N_p$. This is done by assigning a max-plus binary variable for each of the combination of jobs/cycles as:

$$\begin{aligned} z_5(k+i) &\geq z_5(k+j) + p_5 + s_{i,j}(k+i) \\ z_5(k+i) &\geq z_5(k+j) \otimes p_5 \otimes s_{i,j}(k+i) \end{aligned} \tag{4-11}$$

Where $i \neq j$ and $s_{i,j}(k+1) = 0$ if $z_5(k+i)$ is scheduled after $z_5(k+j)$ and $s_{i,j}(k+1) = \varepsilon$ if $z_5(k+i)$ is scheduler before $z_5(k+j)$. As will be later elaborated, a lot of these max-plus binary variables of different cycles/batches are the conjunct of max-plus binary variables of other cycles. In other words, if for instance job 1 is scheduled after job 4, then automatically job 4 is scheduler before job 1. Continuing the step in equation 4-11 for all possible combinations of jobs from different cycles, the constraints are obtained for the ordering process.

By looking closely at the derivations above, only maximizations are obtained. This will make the states $z(k)$ be chosen above a certain threshold, but not closely to this threshold. In other words, the states can be chosen arbitrarly above the threshold. Of course, this is not desired since we want the jobs in every cycle to be finished as fast as possible. This is done by assigning the constraints to a certain cost function, also described in chapter 2 about model predictive control. The cost function initially will function to minimize every job as fast as possible, and maximizing the inputs such that the delivered batches come in as late as possible but the finished batches are obtained as fast as possible. This is done by assigning the following cost function.

$$J(k) = \sum_{j=0}^{Np} y(k+j) - \sum_{j=0}^{N_p} u(k+j) \tag{4-12}$$

The output of the system $y(k)$ is given by the simple max-plus equation:

$$\begin{aligned} y(k) &= C \otimes z(k) \\ y(k) &= \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & p_5 \end{bmatrix} \otimes z(k) \end{aligned} \tag{4-13}$$

Essentially this means the cost function will boil down to:

$$J(k) = \sum_{j=0}^{Np}(z_5(k+j) + p_5) - \sum_{j=0}^{N_p} u(k+j) \tag{4-14}$$

In addition, also scaling factors can be implemented in the cost function for the output and input. This is done when preferable specific input batches or output batches are of a higher concern than others. For instance, important customers can never have delayed jobs. The above problem can be considered as the most basic control problem for model predictive scheduling. However though, the constraints are all defined in max-plus algebra and more importantly max-plus binary variables. In the next section, we observe how the above constraint matrices and cost function are translated to a Mixed-Integer Linear Programming (MILP) framework. Additional constraints are also added in this framework for the sake of feasibility. In the sections about results, we will also see how the cost function is adapted to certain results obtained.

4-1-2 Mixed-integer linear programming

Mixed-Integer linear programming(MILP) problems can simply be seen as problems in the format:

$$\begin{gathered} \min_{\tilde{q}} c^T q \\ \text{s.t.} \\ Aq = b \\ Cq \leq d \\ lb \leq q \leq ub \end{gathered} \tag{4-15}$$

Where the constraints can be equalities, inequalities and bounds on the objective q, and where the cost function is a linear function in the vector q. q is a vector containing all variables contained in the optimization which are integers or just normal numbers. lb and ub are lower- and upperbounds on the variables in q. Note that an integer with lowerbound 0 and upperbound 1 can be seen as normal binary variable. In the following paragraphs, it is shown how the case study of the beer brewery is fitted in this framework with the defined constraints described in the previous section, and additional constraints. It is also shown how constraints defined in max-plus algebra are translated to constraints in the MILP-framework. First the cost function is defined, afterwards the constraints with respect to the state are derived, and at last additional constraints are elaborated.

Cost function

The minimization is the cost function of the problem. In this case the cost function is described in 4-12. In this problem, the vector q will consist of all the states, the binary variables for

the routing, the input variables, and the binary variables for the ordering:

$$q = \begin{bmatrix} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{bmatrix} \tag{4-16}$$

Note that for the variables $z(k)$ and $v(k)$ in q, also the variable for cycle $k-1$ is considered. This is done just for possible initialization receiving information from past cycles. In this study, they are taking into account in the MILP-problem, but will not influence the problem by taking them equal to 0. Note that each of the states $z(k)$ are vectors of length 5 because each state has $z_1(k), ..., z_5(k)$ the states of the 5 machines included. The length of the routing binary variables $v(k)$ is equal to 4, because each $v(k)$ includes $v_1(k), ..., v_4(k)$ for each of the modes/routes per cycle/batch k. The length of the input variables $u(k)$ is 1 for each $u(k)$. The binary variables with respect to the ordering are a little bit different. As we will see later on, the binary variables of future cycles are the conjuncts of binary variables in previous cycles. The length of the vector containing all $s_{i,j}(k)$ therefore equals $\sum_{n=1}^{Np} n$. The length of the total vector therefore equals $5 \times (N_p + 2) + 4 \times (N_p + 2) + (N_p + 1) + \sum_{n=1}^{N_p} n$. In this case study, $N_p = 7$ such that $z(k)$ till $z(k+7)$ exactly ends at $z(8)$ when starting at $k = 1$. This can be visualised by an 8 amount batch order. The length of the vector q in the optimization therefore equals 117×1. To give a better overview of the function and matrices, the cost function can be divided in 4 separate matrices:

$$\min_{\tilde{q}} c^T \tilde{q} = \min_{\tilde{z},\tilde{v},\tilde{u},\tilde{s}} c_1^T \tilde{z} + c_2^T \tilde{v} + c_3^T \tilde{u} + c_4^T \tilde{s} \tag{4-17}$$

The cost function from 4-12 can be expressed within the matrix c^T by using equation 4-13 such that:

$$y(k) = C \otimes z(k) \tag{4-18}$$

Such that the cost function will look like:

$$\min_{\tilde{z},\tilde{v},\tilde{u},\tilde{s}} \underbrace{\begin{bmatrix} 0 & H & \dots & H|0 & \dots & 0|\kappa_1 & \dots & \kappa_{N_p}|0 & \dots & 0 \end{bmatrix}}_{c^T} \underbrace{\begin{bmatrix} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{bmatrix}}_{q} + \sum_{n=1}^{N_p+1} p_5 \tag{4-19}$$

With matrix H as:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-20}$$

Such that only $z_5(k), ..., z_5(k+N_p)$ is captured from every state $z(k), ..., z(k+N_p)$. The term $\sum_{n=1}^{N_p+1} p_5$ contributes to the value of the cost function, but does not influence the minimization since it is considered as a constant value or not a value which can be influenced. Therefore it can be left out. Later on, we will see that also on the variables $v(k), ..., v(k+N_p)$ a cost can be added such they switch only if it is necessary to a certain degree. The optimization variables $z(k-1)$ and $v(k-1)$ are taken into account as some fixed initial values which can be maximized on, but will not be taken into account in the case-study, i.e. will be taken as 0. Furthermore, the parameters κ are taking as potential scaling factors of the input time-instances. However, as will be later on substantiated, all input time-instances $u(k), ..., u(k+N_p)$ will be taken equal to 0.

Constraints w.r.t. the routing of the scheduler

The cost function is derived in the MILP-framework. Now the constraints need to be converted to this framework. The equations for $z(k+N_p)$ can be boiled down to $z(k)$ and $z(k-1)$ by iteratively using the system dynamics. This is fortunately not necessary because the vector q in the MILP-problem consists of all the states at each event-step k as can be seen in equation 4-16. Next, one derivation is made for the event with respect to its state $z(k)$, where as for all the other events, these follow directly. For the equations of 4-7 for event-step $z(k)$ the

following equation can be obtained:

$$\begin{aligned} z(k) \geq & \bigoplus_{\ell(k)=1}^{4} (v_\ell(k) \otimes A_0(\ell(k))) \otimes z(k) \\ & \bigoplus_{\ell(k-1)=1}^{4} (v_\ell(k-1) \otimes A_1(\ell(k-1)) \otimes z(k-1) \\ & \bigoplus_{\ell(k)=1}^{4} (v_\ell(k) \otimes B(\ell(k)) \otimes u(k) \end{aligned} \tag{4-21}$$

Ultimately this is the same as constraining all the maximizations in a greater or equal constraint with respect to the state $z(k)$. The following derivations for the constraints are only made for $z(k)$:

$$z(k) \geq \left[\begin{array}{c} v_1(k) \otimes A_0(\ell=1) \otimes z(k) \\ v_2(k) \otimes A_0(\ell=2) \otimes z(k) \\ v_3(k) \otimes A_0(\ell=3) \otimes z(k) \\ v_4(k) \otimes A_0(\ell=4) \otimes z(k) \\ \hline v_1(k-1) \otimes A_1(\ell=1) \otimes z(k-1) \\ v_2(k-1) \otimes A_1(\ell=2) \otimes z(k-1) \\ v_3(k-1) \otimes A_1(\ell=3) \otimes z(k-1) \\ v_4(k-1) \otimes A_1(\ell=4) \otimes z(k-1) \\ \hline v_1(k) \otimes B(\ell=1) \otimes u(k) \\ v_2(k) \otimes B(\ell=1) \otimes u(k) \\ v_3(k) \otimes B(\ell=1) \otimes u(k) \\ v_4(k) \otimes B(\ell=1) \otimes u(k) \end{array}\right] \tag{4-22}$$

Repeating this step for every state $z(k)$ to $z(k+N_p)$ and stacking these into one matrix, one can obtain a big constraint matrix. Since the binary variables are still defined in max-plus sense, they need to be translated to normal binary variables for the MILP-solver. The translation is given in the chapter 2 in equation 2-9. Though in this equation, ε is defined as infinity. Since computer programmes often do not like to work with infinite numbers, a very large number β is used instead such that:

$$\begin{aligned} \beta << 0 \\ v_\beta(k) = \beta(1 - v(k)) \end{aligned} \tag{4-23}$$

With each of the maximizations of 4-22 the following can be obtained:

$$\begin{aligned} z(k) \geq v_{1,\varepsilon}(k) \otimes & \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ p_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & p_3 & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} \\ & = v_{1,\varepsilon}(k) \otimes \begin{bmatrix} \varepsilon \\ \varepsilon \\ p_1 + z_1(k) \\ \varepsilon \\ p_3 + z_3(k) \end{bmatrix} \end{aligned} \tag{4-24}$$

Now by replacing the max-plus binary variable $v_{1,\varepsilon}(k)$ by a normal binary variable $\beta(1-v_1(k))$, and working out the max-plus operators to conventional algebra, as well as replacing the values for ε by β, the following equation can be obtained.

$$z(k) \geq \begin{bmatrix} \beta \\ \beta \\ p_1 + z_1(k) + \beta - \beta v_1(k)) \\ \beta \\ p_3 + z_3(k) + \beta - \beta v_1(k) \end{bmatrix} \tag{4-25}$$

The inequalities can be reformulated as:

$$z(k) \geq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\beta \\ 0 \\ -\beta \end{bmatrix} v_1(k) + \begin{bmatrix} \beta \\ \beta \\ p_1 + \beta \\ \beta \\ p_3 + \beta \end{bmatrix} \tag{4-26}$$

Continuing this process for the other modes regarding A_0 in the constraint of 4-22, the following constraints can be added:

$$z(k) \geq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\beta \\ -\beta \end{bmatrix} v_2(k) + \begin{bmatrix} \beta \\ \beta \\ \beta \\ p_1 + \beta \\ p_4 + \beta \end{bmatrix}$$

$$z(k) \geq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\beta \\ 0 \\ -\beta \end{bmatrix} v_3(k) + \begin{bmatrix} \beta \\ \beta \\ p_2 + \beta \\ \beta \\ p_3 + \beta \end{bmatrix} \tag{4-27}$$

$$z(k) \geq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\beta \\ -\beta \end{bmatrix} v_4(k) + \begin{bmatrix} \beta \\ \beta \\ \beta \\ p_2 + \beta \\ p_4 + \beta \end{bmatrix}$$

Now the procedure is done again for the second set of inequalities from 4-22. These constrain the current state with respect to the matrix previous state by the matrix A_1. Again the derivations are shown for only the first state $z(k)$, where the constraints for the other states

follow directly. The following inequality can be formulated:

$$z(k) \geq v_{1,\varepsilon}(k-1) \otimes \begin{bmatrix} p_1 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & p_3 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \\ z_3(k-1) \\ z_4(k-1) \\ z_5(k-1) \end{bmatrix} \tag{4-28}$$

$$= \begin{bmatrix} p_1 + z_1(k-1) + \beta - \beta v_1(k-1) \\ z_2(k-1) + \beta - \beta v_1(k-1) \\ p_3 + z_3(k-1) + \beta - \beta v_1(k-1) \\ z_4(k-1) + \beta - \beta v_1(k-1) \\ \beta - \beta v_1(k-1) \end{bmatrix}$$

Or in another notation:

$$z(k) \geq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \\ z_3(k-1) \\ z_4(k-1) \\ z_5(k-1) \end{bmatrix} + \begin{bmatrix} -\beta \\ -\beta \\ -\beta \\ -\beta \\ -\beta \end{bmatrix} v_1(k-1) + \begin{bmatrix} p_1 + \beta \\ \beta \\ p_3 + \beta \\ \beta \\ \beta \end{bmatrix} \tag{4-29}$$

As for the other modes:

$$z(k) \geq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \\ z_3(k-1) \\ z_4(k-1) \\ z_5(k-1) \end{bmatrix} + \begin{bmatrix} -\beta \\ -\beta \\ -\beta \\ -\beta \\ -\beta \end{bmatrix} v_2(k-1) + \begin{bmatrix} p_1 + \beta \\ \beta \\ \beta \\ p_4 + \beta \\ \beta \end{bmatrix}$$

$$z(k) \geq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \\ z_3(k-1) \\ z_4(k-1) \\ z_5(k-1) \end{bmatrix} + \begin{bmatrix} -\beta \\ -\beta \\ -\beta \\ -\beta \\ -\beta \end{bmatrix} v_3(k-1) + \begin{bmatrix} \beta \\ p_2 + \beta \\ p_3 + \beta \\ \beta \\ \beta \end{bmatrix} \tag{4-30}$$

$$z(k) \geq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \\ z_3(k-1) \\ z_4(k-1) \\ z_5(k-1) \end{bmatrix} + \begin{bmatrix} -\beta \\ -\beta \\ -\beta \\ -\beta \\ -\beta \end{bmatrix} v_4(k-1) + \begin{bmatrix} \beta \\ p_2 + \beta \\ \beta \\ p_4 + \beta \\ \beta \end{bmatrix}$$

For the last part of 4-22, regarding the constraints for $z(k)$ with respect to the input, the following derivations can be made:

$$z(k) \geq v_{1,\varepsilon}(k) \otimes \begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix} \otimes u(k) = \begin{bmatrix} u(k) + \beta - \beta v_1(k) \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix} \tag{4-31}$$

Or in another notation:

$$z(k) \geq \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} u(k) + \begin{bmatrix}-\beta\\0\\0\\0\\0\end{bmatrix} v_1(k) + \begin{bmatrix}\beta\\\beta\\\beta\\\beta\\\beta\end{bmatrix} \tag{4-32}$$

Continuing this for the other modes this will give:

$$\begin{aligned} z(k) &\geq \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} u(k) + \begin{bmatrix}-\beta\\0\\0\\0\\0\end{bmatrix} v_2(k) + \begin{bmatrix}\beta\\\beta\\\beta\\\beta\\\beta\end{bmatrix} \\ z(k) &\geq \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} u(k) + \begin{bmatrix}0\\-\beta\\0\\0\\0\end{bmatrix} v_3(k) + \begin{bmatrix}\beta\\\beta\\\beta\\\beta\\\beta\end{bmatrix} \\ z(k) &\geq \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} u(k) + \begin{bmatrix}0\\-\beta\\0\\0\\0\end{bmatrix} v_4(k) + \begin{bmatrix}\beta\\\beta\\\beta\\\beta\\\beta\end{bmatrix} \end{aligned} \tag{4-33}$$

By setting up the matrices for each event-step k till $k + N_p$, all states $z(k)$ up till $z(k + N_p)$ will be lower bounded for the routing process, i.e. the states must be higher or equal to this lowerbound. By minimizing these greater or equal constraints due to the cost function, the overal behaviour of the state $z(k)$ will behave as the optimal max-plus solution w.r.t. the routing of the prescribed cost function.

Constraints w.r.t. the ordering of the scheduler

In the ordering process of the scheduler, only machine 5 is considered. Since all jobs need to go through the process of machine 5, all jobs need to be ordered with respect to each other accordingly. This is different for instance, when looking at one of the other machines, since not all routes go through a specific machine, only the cycles in which the routes are chosen to go through the specific machine need to be ordered accordingly. The ordering is modeled as follows. To start with the first cycle k until cycle $k + N_p$. The ordering with respect to the processes in machine 5 of the other cycles can be modelled, as is already mentioned, by the max-plus binary variables $s_{i,j}(k)$. This is done by making a relation of the starting time of process 5 in machine 5 of cycle k with respect to the other cycles. As an example, we take cycle k and cycle $k + 1$. We define the order as:

$$z_5(k) \geq z_5(k + 1) \otimes p_5(k + 1) \otimes s_{0,1}(k) \tag{4-34}$$

If we choose the job in cycle $k+1$ to go before the job in cycle k to go in the process of machine 5, we choose $s_{0,1}(k) = 0$, because the equation then becomes:

$$z_5(k) \geq z_5(k+1) \otimes p_5(k+1) \tag{4-35}$$

If we choose the job in cycle $k+1$ not to go before the job in cycle k to go in machine 5, we choose $s_{0,1} = \varepsilon$ since the equation then becomes:

$$z_5(k) \geq z_5(k+1) \otimes p_5(k+1) \otimes \varepsilon = \varepsilon \tag{4-36}$$

Which means $z_5(k) \geq \varepsilon$ which is always the case. Though, the above equation only mentions that $z_5(k+1)$ is not before $z_5(k)$, not necessarily that $z_5(k)$ is before $z_5(k+1)$ by taking $s_{0,1}(k) = \varepsilon$. However, by making that statement, it is clear that this is the case. This is modelled by setting the binary variable of the maximization of $z_5(k+1)$ with respect to $z_5(k)$ as the conjunct of the binary variable from the maximization of $z_5(k)$ with respect to $z_5(k+1)$. Mathematically formulated:

$$z_5(k+1) \geq z_5(k) \otimes p_5(k) \otimes s_{1,0}(k+1) \tag{4-37}$$

Where $s_{1,0}(k+1) = \bar{s}_{0,1}(k)$. If we now derive all the maximization of $z_5(k)$ with respect to $z_5(k+1), ..., z_5(k+N_p)$ we obtain:

$$\begin{aligned} z_5(k) &\geq z_5(k+1) \otimes p_5(k+1) \otimes s_{0,1}(k) \\ z_5(k) &\geq z_5(k+2) \otimes p_5(k+2) \otimes s_{0,2}(k) \\ &\vdots \\ z_5(k) &\geq z_5(k+N_p) \otimes p_5(k+N_p) \otimes s_{0,N_p}(k) \end{aligned} \tag{4-38}$$

The same procedure for $z_5(k+1)$ with respect to $z_5(k)$ and $z_5(k+2), ..., z_5(k+N_p)$:

$$\begin{aligned} z_5(k+1) &\geq z_5(k) \otimes p_5(k) \otimes s_{1,0}(k+1) \\ z_5(k+1) &\geq z_5(k+2) \otimes p_5(k+2) \otimes s_{1,2}(k+1) \\ &\vdots \\ z_5(k+1) &\geq z_5(k+N_p) \otimes p_5(k+N_p) \otimes s_{1,N_p}(k+N_p) \end{aligned} \tag{4-39}$$

Or in general:

$$z_5(k+i) \geq z_5(k+j) \otimes p_5(k+j) \otimes s_{i,j}(k+i) \tag{4-40}$$

For $i \neq j$ and i, j containing values from 0 to N_p. The conjunct of the binary variables is modelled as:

$$s_{i,j}(k+i) = \bar{s}_{j,i}(k+j) \tag{4-41}$$

If we now want to implement the constraint in the MILP-problem, the following can be concluded. First of all lets have a look at the vector:

$$q = \begin{bmatrix} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{bmatrix} \tag{4-42}$$

The length of the bottom part $s_{0,i}(k)$ up till $s_{N_p,i}$ is equal to $\sum_{n=1}^{N_p} n$ instead of N_p^2 since for each cycle $k+1$ up to $k+N_p$ there can be made one more conjunct with respect to the maximizations of the previous cycles. Therefore it boils down to possible new binary variables as $N_p + (N_p - 1) + (N_p - 2) + ... + 1 = \sum_{n=1}^{N_p} n$.

To put the constraints in the MILP-framework, we first have to rewrite the constraints. As an example, lets take equation 4-34. This can be converted to conventional algebra as:

$$z_5(k) \geq z_5(k+1) + p_5(k+1) + s_{0,1}(k) \tag{4-43}$$

The max-plus binary variable converted as is done in equation 2-9, followed by rewriting the equation in the format $Fq \leq b$ gives the following derivations. The derivations for one of the constraints for the state $z_5(k)$ are given by:

$$\begin{aligned} z_5(k) &\geq z_5(k+1) + p_5(k+1) + \beta(1 - s_{0,1}(k)) \\ z_5(k) &\geq z_5(k+1) + p_5(k+1) + \beta - \beta s_{0,1}(k) \\ \underbrace{-z_5(k) + z_5(k+1) - \beta s_{0,1}(k)}_{F_{ord_{0,1}} q} &\leq \underbrace{-\beta - p_5(k+1)}_{b_{ord_{0,1}}} \end{aligned} \tag{4-44}$$

For the constraints for the state $z_5(k+1)$, the first constraint is the conjunct of the binary variable described in equation 4-44, given by:

$$\begin{aligned} z_5(k+1) &\geq z_5(k) + p_5(k) + \beta(1 - s_{1,0}(k+1)) \\ z_5(k+1) &\geq z_5(k) + p_5(k) + \beta(1 - (1 - s_{0,1}(k))) \\ z_5(k+1) &\geq z_5(k) + p_5(k) + \beta(s_{0,1}(k)) \\ \underbrace{z_5(k) - z_5(k+1) + \beta(s_{0,1}(k))}_{F_{ord_{1,0}}q} &\leq \underbrace{-p_5(k)}_{b_{ord_{1,0}}} \end{aligned} \tag{4-45}$$

Continuing the process of equations 4-38 and 4-39 for all states $z_5(k)$ with respect to the events $k, ..., k+N_p$ and replacing the unnecessary binary variables with their conjunct binary variables from previous cycles as is done in equation 4-45, all constraints with respect to the ordering process in machine 5 are obtained. Stacking these into one big matrix will give with appropriate positions with respect to the vector 4-42 gives:

$$\underbrace{\begin{bmatrix} F_{ord_{0,1}} \\ \vdots \\ F_{ord_{0,N_p}} \\ F_{ord_{1,0}} \\ \vdots \\ F_{ord_{N_p,N_p-1}} \end{bmatrix}}_{F_{ord}} \begin{bmatrix} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{bmatrix} \leq \underbrace{\begin{bmatrix} b_{ord_{0,1}} \\ \vdots \\ b_{ord_{0,N_p}} \\ b_{ord_{1,0}} \\ \vdots \\ b_{ord_{N_p,N_p-1}} \end{bmatrix}}_{b_{ord}} \tag{4-46}$$

Additional constraints w.r.t. inputs and binary variables

Now the constraints for the states are obtained, additional constraints need to be added. This is to prevent infeasible solutions. The binary variables need to be constrained due to the fact only one mode per event-step k can be chosen. As well, the input time-instances need to be constrained, since input time-instances in the future cycles can never be less than the input time-instance from the present cycle. In our case, the input is taken as 0 for all cycles, however for a general approach these constraints are still added. All these additional constraints are given below, to begin with constraining the input:

$$\begin{aligned} u(k+j-1) &\leq u(k+j) \\ u(k+j-1) - u(k+j) &\leq 0 \end{aligned} \tag{4-47}$$

Such that a inequality matrix can be made:

$$\underbrace{\left[\begin{array}{ccccccccccc} 0 & \dots & 0|0 & \dots & 0|1 & -1 & 0 & \dots & 0|0\dots & 0 \\ 0 & & 0|0 & \dots & 0|0 & 1 & -1 & \dots & 0|0\dots & 0 \\ \vdots & & \vdots & & & & \ddots & & & \\ 0 & \dots & 0|0 & \dots & 0|0 & 0 & \dots & 1 & -1|0\dots & 0 \end{array}\right]}_{F_u} \left[\begin{array}{c} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{array}\right] \leq \underbrace{\left[\begin{array}{c} 0 \\ \vdots \\ 0 \end{array}\right]}_{b_u} \tag{4-48}$$

Furthermore, the max-plus binary variables need to be constrained, such that only one modes is chosen per event-step k. Fortunately, by translating the max-plus binary variables to normal binary variables, constraints in the MILP-framework will work as well for constraining the normal binary variables. Constraining the binary variables with lowerbound 0 and upperbound 1, makes them in the MILP-framework behave like binary variables. Thereafter contraining them with respect to each other as:

$$v_1(k) + v_2(k) + v_3(k) + v_4(k) = 1 \tag{4-49}$$

Now only one mode per event step can be chosen. Continuing this step for alle cycles k will make the following constraints:

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\underbrace{\left[\begin{array}{ccccccccccc} 0 & \dots & 0|T & 0 & \dots & 0|0 & \dots & 0|0\dots & 0 \\ 0 & & 0|0 & T & 0\dots & 0|0 & \dots & 0|0\dots & 0 \\ & \vdots & & & \ddots & & & & \\ 0 & \dots & 0|0 & \dots & 0 & T|0 & \dots & 0|0\dots & 0 \end{array}\right]}_{V_{eq}} \left[\begin{array}{c} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{array}\right] = \underbrace{\left[\begin{array}{c} 1 \\ \vdots \\ 1 \end{array}\right]}_{b_{eq}} \tag{4-50}$$

Finally note that this last constraint matrix is an equality constraint matrix instead of an inequality constraint matrix, as is done for the constraint matrices before.

4-1-3 Model predictive scheduling

Statical Scheduler

The statical scheduler can be obtained by just minimizing the initial cost function subjected to the constraint described in the above paragraphs. With the process times p_i for $i = 1, .., 5$ fixed for all cycles k, the outcome will be a sequence of modes which is already determined at the initial optimization. In other words, no additive information is given to the system, and all information is already fixed when optimizing.

One thing to notice however, is the sake of a unique solution. Since the system is rather simple, multiple solutions can come out of the optimization which are all optimal for the statical scheduler. For the statical scheduler this is not a problem since the schedule is fixed at the initial time-instance, however for the model predictive scheduler, it is not wanted the adaptive scheduler switches arbitrarily between optimal sequences of modes, still it needs to if the benefits are sufficiently high. To avoid the unnecessary switching between optimal modes, in the cost function the inner product is added between the previous and currently to be chosen modes:

$$\min_{\tilde{z},\tilde{v},\tilde{u},\tilde{s}} \underbrace{\begin{bmatrix} 0 & H & \dots & H | -av(k-1) & \dots & -av(k+N_p) | \kappa_1 & \dots & \kappa_{N_p} | 0 & \dots & 0 \end{bmatrix}}_{c^T} \underbrace{\begin{bmatrix} z(k-1) \\ z(k) \\ \vdots \\ z(k+N_p) \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline u(k) \\ \vdots \\ u(k+N_p) \\ \hline s_{0,i}(k) \\ \vdots \\ s_{N_p,i}(k+N_p) \end{bmatrix}}_{q} + \sum_{n=1}^{N_p+1} p_5 \tag{4-51}$$

Where $H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ and a is a small number just above 0. The parameter a is chosen sufficiently small such that it does not have an influence to the overal cost function,

but has an influence on unnecessary switching between optimal sequences of modes. Now the updated scheduler will multiply the previous modes with the modes it tries to find for the new situation. If the modes are the same, it will have a negative impact on the minimization in the cost function, and therefore if there are multiple optimal mode sequences, it will remain with the previous mode sequence. Only when it has a beneficial impact on the cost function, bigger than the beneficial impact of remaining in the same modes, it changes from mode sequences.

As can be concluded, the statical scheduler can already give a lot of insight information about the system, and can be handled as an initial schedule for the dynamical schedulers. Furthermore, the statical scheduler need to be obtained to compare results with the dynamical schedulers. All the results will not be discussed in this chapter, but in the next chapter.

Event-based update cycle k

The dynamical scheduler is a schedule which updates by getting new information about some of its processes. This new information is obtained by receiving the expected ending times of the extended Kalman filters. In the event-based update, these ending times can be considered per cycle k. In other words, at each cycle k, the optimization is executed, where the model predictive scheduler fills in the necessary solutions and continues to the next cycle $k+1$, where new information about the process times $p_i(k+1)$ is obtained.

The information the system receives, can be constructed in the vector $p_i(k)$, such that the process times become variable within every cycle k. Simply reconstructing the constraint matrices with variable $p_i(k)$ in each of the cycles and updating the scheduler will give the solution. Since the process times only occur in the matrices A_0 and A_1, the reconstruction with variable process times per cycle can be translated to these matrices as:

$$\begin{aligned} A_0(\ell(k)) &= A_0(\ell(k), p_i(k)) \\ A_1(\ell(k-1)) &= A_1(\ell(k-1), p_i(k-1)) \end{aligned} \tag{4-52}$$

For all $k, ..., k+N_p$ and possible modes $\ell(k)$. Reconstructing these into the MILP framework will put the processing times on the side with all the constants. Taking 4-26 as an example, the constraint matrices become:

$$z(k) \geq \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{F(\ell=1)} \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ -\beta \\ 0 \\ -\beta \end{bmatrix}}_{G(\ell=1)} v_1(k) + \underbrace{\begin{bmatrix} \beta \\ \beta \\ \beta \\ \beta \\ \beta \end{bmatrix}}_{\phi} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ p_1(k) \\ 0 \\ p_3(k) \end{bmatrix}}_{\theta(k),\ell=1} \tag{4-53}$$

And so on for the rest of the constraint matrices. Now the variable processing times are implemented, one has to look at how the scheduler is updated when continuing to a next cycle. In other words, how does the scheduler fix values or choices which are already in the passed cycles $k-1, ..., k-\mu_{max}$ The solution lies in the equality constraints of the MILP-solver. Of course the MILP-vector q from equation 4-42 contains the vectors for the modes

$v_i(k)$. When continuing to the cycle $k+1$, all cycles up till k are already chosen and therefore fixed. For example, if we are in cycle $k+1$ and in cycle k the mode $\ell = 1$ is chosen, the equality constraints can be updated by addition of:

$$\begin{bmatrix} 0 & \dots & 0|0 & F_k & 0 & \dots & 0|0 & \dots 0 \end{bmatrix} \begin{bmatrix} \tilde{z} \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline \tilde{u} \\ \tilde{s} \end{bmatrix} = b_k \tag{4-54}$$

With F_k and b_k chosen accordingly, since mode 1 is chosen in cycle k:

$$F_k = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \quad b_k = 1 \tag{4-55}$$

Subsequently, continuing this step for the next cycle $k+1$ by optimizing the cost function over the cycles $k+1$ up till $k+N_p$ gives a new mode sequence, where the mode for cycle k is fixed as $\ell(k) = 1$ and therefore $v(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$.By receiving the next optimal mode sequence and constructing the new equality constraint with matrices F_{k+1} and b_{k+1} fixes the modes up till $k+1$. Now continuing this step for cycles $k+2$ up till $k+N_p$, and so on. As we will see in the next chapter, problems will arise when updating the scheduler with an event-based update. In the event-based updated scheduler, the information is received from the previous cycle when entering the current cycle. In other words, for instance when machine 1 in cycle k happens to have a much larger process time, this information is received in cycle $k+1$, such that the schedule can adapt to this deviation in process time from the previous cycle.

Time-based update discrete time-instant κ

In the previous paragraph, the model predictive scheduler is updated in the event-based domain. In this paragraph, the scheduler is assumed to update in the time-based domain. Of course, this is much harder since the scheduler itself is constructed in the event-based domain. The translation from the time-domain to the event-domain and vice versa is explained in the next content. First of all the matrices with respect to the discrete time-index κ can be changed as:

$$\begin{aligned} A_0(\ell(k)) &= A_0(\ell(k), p_i(k,\kappa)) \\ A_1(\ell(k-1)) &= A_1(\ell(k-1), p_i(k-1,\kappa)) \end{aligned} \tag{4-56}$$

Such that the processing times in cycle k are not only dependant on cycle k but also on the discrete time-counter κ. An illustrative example, such that the meaning of $p(k,\kappa)$ becomes a little bit more clear. For instance, when in the current cycle k, the mode is chosen to be $\ell(k) = 1$. The initial optimization now finds the starting times for the processes in cycle k. Since the mode is equal to 1, the machine order will be $M_1 \rightarrow M_3 \rightarrow M_5$. Suppose the starting time of machine M_1 in cycle k is equal to 0, i.e. $z_1(k) = 0$ and the initial processing time is equal to 7 time-units. The estimated end time $p_1(k,0)$ is of course 7 when considering $\kappa = 0$. By updating the time-step with 1, such that we are at $\kappa = 1$ the process $p_1(k,1)$

is probably still busy since it is very unlikely it has already finished its job with the initial process time of $p_1(k,0) = 7$. Though, after 1 time-step, new information comes in from the measurements such that the extended Kalman filter can make a new prediction of the end-time. Suppose this new end-time is way faster than the 7 time-units, or way slower than the initial 7 time-units, one could adapt the routing of all the jobs for a faster end time of the overal cost function for the scheduler.

Another problem which arises, is that a certain mode can change inside the cycle. In the event-based scheduler which updated with respect to the cycle k, discussed in the previous paragraph, the updated schedule gives an updated mode change for the whole cycle, such that per cycle a new mode or the same mode is chosen, and so on to the next cycle. The mode in the previous cycle is then fixed and can not be changed anymore. In the time-based update however, it could happen that a certain mode which is already started, could still be changed in the cycle itself. To visualize this, an example is given. If again in cycle k , mode $\ell(k) = 1$ is chosen, and the starting time of machine 1 will be $z_1(k) = 0$ with processing time $p_1(k,0) = 7$. Now at the next time-instant $\kappa = 1$, machine 1 is already active, such that machine 1 must be contained in the mode for cycle k. Though, machine 3, is not yet started since it has a starting time $z_3(k) \geq z_1(k) + p_1(k,0)$. If time continues to $\kappa = 1, 2, ...$ it could happen that the process time of machine 1 in cycle k is much less or much more. The route is not yet fixed for machine 3, therefore in cycle k the modes can be adjusted with respect to the machines 3 and 4, namely $M_1 \to M_3 \to M_5$ or $M_1 \to M_4 \to M_5$.

The way this is modelled in the model predictive scheduler is as follows. From the initial schedule the starting times are obtained($\kappa = 0$), as well as the modes chosen for each of the cycles. These starting-times and modes are given as input for the next time-instant $\kappa = 1$. If the time-instant is larger than the starting time of a specific process, the process is irreversible and considered fixed. To model this, actually the opposite of the approach in the cyclic update is proposed. Namely, if $z_1(k)$ already started due to the fact the time-counter κ is passed the starting time of $z_1(k)$, one can not conclude a specific mode has been chosen yet. What can be concluded, is that specific modes are not chosen. In case of $z_1(k)$ is started, it can be concluded that modes 3 and 4 can not be chosen anymore, because mode 3 and 4 start in machine $z_2(k)$. Mathematically this can be done by adding equality constraints such that mode 3 and 4 can not be chosen for cycle k:

$$\begin{bmatrix} 0 & \dots & 0|0 & F_k & 0 & \dots & 0|0 & \dots 0 \end{bmatrix} \begin{bmatrix} \tilde{z} \\ \hline v(k-1) \\ v(k) \\ \vdots \\ v(k+N_p) \\ \hline \tilde{u} \\ \tilde{s} \end{bmatrix} = b_k \tag{4-57}$$

With the zero-values represent appropriate size zero-matrices. Now if we take the the matrices F_k and b_k as:

$$F_k = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad b_k = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4-58}$$

This mean the binary vector $v(k)$ can only, but also must at least and at most, have values 1 for either the position where $v_1(k) = 1$ or $v_2(k) = 1$. This is concluded for the reason of the equality constraint already mentioned previously:

$$v_1(k) + v_2(k) + v_3(k) + v_4(k) = 1 \tag{4-59}$$

With this framework continued when machine 3 and 4 are started, namely fixing the modes which can not be chosen anymore when machine 3 and 4 are started, it automatically chooses one of the modes per cycle when the machine is done in all machines for a specific cycle.

4-2 Subsystems

In this model, a beer brewing factory is taken as case study. The whole process of beer brewing is strongly simplified. Essentially, beer brewing consists of a lot of steps, from filtering or clarification in intermediate steps, to the milling of the grist to starch before the mashing etc. In this case study, the process is simplified to only 3 processes; Mashing, brewing and fermentation. The factory features 2 mashing tanks, 2 brewing tanks and 1 fermentation tank. Each batch need to undergo all 3 processes in consecutive order. State-space models are obtained, where simulations are made to obtain realizations and data of the processes. The fictitious different mashing and brewing tanks will also vary in processing times, this will let the system vary in processing times per route such that optimality can be obtained. General information about the brewing process of beer can be found in [19]. In the following subsystems, the discrete time-domain is featured for the state-spaces. Since k is already used for the event domain, and to avoid confusion, κ is used as the discrete time-counter.

4-2-1 Mashing

Mashing is one of the key processes in the beer brewing process. It takes care of the larger sugar molecules to boil down to smaller fermentable sugar molecules such that in the later stage, fermentation can start. Modern day, the process of mashing often decides how strong the beer will get with respect to the alcohol percentage due to the sugars obtained in the mashing stage. These fermentable sugars are obtained by enzyms, mostly α- and β-amylases. The process itself is quite complex, due to the fact different enzyms work optimal at different temperatures, as well that these different enzyms produce different smaller sugars. Also the enzyms are not controllable, in the sense one can not add or remove the enzyms since the enzyms themselves are already contained in the malted barley. Moreover, the size and distribution of the larger sugar molecules are dependant on the type of malted barley; In other words, every new batch of malted barley could have different characteristics with respect to mashing process. The conclusion is that model-predictive methods on the end-time of the mashing-process are rather complex and often explained by experimental measures. For that reason, a lot of simplifications and assumptions are made, such that a model is obtained which is reasonable enough to represent the process of mashing, but still not to complex or influenced such that predictions are not possible. The information in this paragraph is mainly discussed in [3].

The first step in mashing is the gelatinisation of the larger starch molecules. The original starch molecules are too large for the enzyms to break them down to smaller fermentable sugars. Thus, gelatinisation is necessary, which means the larger starch molecules by means of temperature are breaking down to still large but smaller sugar molecules, this is called the gelatinisation of the starch. This process is described by a simple first order differential equation:

$$\frac{dS_g(t)}{dt} = k_g e^{-\frac{Eg}{RT}} S_s(t) \tag{4-60}$$

Where $S_s(t)$ is the amount of total starch, E_g the activation energy, R the gas constant, T the temperature and k_g a pre-exponential factor. In this thesis, the gelatinisation is assumed to already have happened, such that the modelling is not of any importance.

In the second stage. The gelatinised starch is respectively transformed into dextrins, maltose, glucose and maltotriose. Thereafter, dextrins can be transformed into maltose, glucose and maltotriose. Maltose, glucose and maltotriose are considered to be fermentable sugars. The step from gelatinised starch and dextrins to maltose can be done by both α- and β-amylases where all the other steps can only be done by α-amylase. In this thesis for the sake of simplicity, it is considered only the process from gelatinised starch to maltose takes place, done by only α-amylase. The process can be mathematically described as:

$$\frac{dS_{mal}(t)}{dt} = r_{mal,\alpha} \tag{4-61}$$

With:

$$r_{mal,\alpha} = k_{mal} a_\alpha S_g(t) \tag{4-62}$$

Where k_{mal} is kinetic factor for the maltose production, a_α is the global activity for the enzym and $S_g(t)$ the gelatinised starch concentration. Considering the process of gelanitisation is already done beforehand, we can consider the state-space model as:

$$\begin{bmatrix} \dot{S}_g(t) \\ \dot{S}_{mal}(t) \end{bmatrix} = \begin{bmatrix} -r_{mal,\alpha} \\ r_{mal,\alpha} \end{bmatrix} \tag{4-63}$$

Replacing $S_g(t)$ and $S_{mal}(t)$ by $x_1(t)$ and $x_2(t)$ we can get the following state-space model.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -k_{mal} a_\alpha x_1(t) \\ k_{mal} a_\alpha x_1(t) \end{bmatrix} \tag{4-64}$$

To make the model more realistic, noise terms are added in the parameter k_{mal}. This is done by assigning an additional state $x_3(t)$ to the behaviour of the parameter as $k_{mal} = k_0 + x_3(t)$. This will result in k_{mal} floating around some value k_0. The state space description with noise can be described as:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} -a_\alpha (k_0 + x_3(t)) x_1(t) \\ a_\alpha (k_0 + x_3(t)) x_1(t) \\ \frac{(\zeta - 1)}{T_s} x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w(t) \tag{4-65}$$

Such that $w(t)$ is a zero-mean white noise signal with covariance $Q(t)$, ζ is an damping coefficient which will be just beneath 1 such that $\zeta \approx 1$ and T_s is the sampling time for

the discretization of the model. The model is not highly non-linear and therefore a simple Euler discretization can be used to get a well approximation. With the sample time T_s as the discrete time-step, the model can be made discrete:

$$\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+2) \\ x_3(\kappa+3) \end{bmatrix} = \begin{bmatrix} x_1(\kappa) - T_s(a_\alpha(k_0 + x_3(\kappa))x_1(\kappa)) \\ x_2(\kappa) + T_s(a_\alpha(k_0 + x_3(\kappa))x_1(\kappa)) \\ \zeta x_3(\kappa) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix} w(\kappa) \tag{4-66}$$

To give more insight to the system w.r.t. the linearizations which need to be done for the estimations, the following state-space description is obtained:

$$\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+2) \\ x_3(\kappa+3) \end{bmatrix} = \begin{bmatrix} x_1(\kappa) - T_s a_\alpha k_0 x_1(\kappa) - T_s a_\alpha x_3(\kappa) x_1(\kappa) \\ x_2(\kappa) + T_s a_\alpha k_0 x_1(\kappa) + T_s a_\alpha x_3(\kappa) x_1(\kappa) \\ \zeta x_3(\kappa) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix} w(\kappa) \tag{4-67}$$

The data received from the measurements, can be described by:

$$y(\kappa) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \\ x_3(\kappa) \end{bmatrix} + v(\kappa) \tag{4-68}$$

4-2-2 Brewing

Brewing is one of the key processes which gives the beer a bitter taste. The bitter taste is mainly caused by the hops "Humulus Lupus" which are added. The value of the hops is mostly determined by the alpha-acids, essential hop oils and polyphenols[11]. In this thesis, the focus will be on the forming of these alpha-acids, since it is considered they take the most part in the bitter flavour. Also, the model does not need to be very complex, since the use of it in the scheduler is more important than the realistic behaviour of the model itself, as is suggested for all the processes in the subsystems. For this reason, the degredation of iso-acids is not taken into account. The brewing can be described by a simple first-order differential equation:

$$\begin{aligned} \tfrac{dC_{\text{alpha-acid}}(t)}{dt} &= -b_{\text{brew}} C_{\text{alpha-acids}}(t) \\ \tfrac{dC_{\text{iso-alpha-acid}}(t)}{dt} &= b_{\text{brew}} C_{\text{alpha-acids}}(t) \end{aligned} \tag{4-69}$$

Where the first terms before the equality sign are the rates in which the concentrations of the alpha-acids and iso-alpha-acids decrease or increase, b_{brew} is a reaction rate parameter which is obtained by taking the Arrhenius equation but considered to be constant, and $C_{\text{alpha-acids}}$ the concentration of the alpha-acids at time t. By implementing the same procedure as done in the mashing tank, an additional state is constructed in which noise is added. By replacing the rate in which the alpha-acids and iso-alpha-acids increase or decrease by x_1 and x_2, and the parameter b_{brew} to be chosen as $b_{\text{brew}} = b_0 + x_3(t)$ the following state-space construction can be obtained:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} -(b_0 + x_3(t))x_1(t) \\ (b_0 + x_3(t))x_1(t) \\ \frac{(\zeta-1)}{T_s} x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w(t) \tag{4-70}$$

Where $w(t)$ is a zero-mean white noise signal, T_s the sampling time, and ζ a factor close to 1 but just beneath it. Now following the same step in the mashing process by taking the discretizations, one can obtain:

$$\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+1) \\ x_3(\kappa+1) \end{bmatrix} = \begin{bmatrix} x_1(\kappa) - T_s b_0 x_1(\kappa) - T_s x_3(\kappa) x_1(\kappa) \\ x_2(\kappa) + T_s b_0 x_1(\kappa) + T_s x_3(\kappa) x_1(\kappa) \\ \zeta x_3(\kappa) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix} w(\kappa) \tag{4-71}$$

Where the output signal is considered to be the level of alpha-acids with the addition of measurement noise:

$$y(\kappa) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \\ x_3(\kappa) \end{bmatrix} + v(\kappa) \tag{4-72}$$

4-2-3 Fermentation

The fermentation process is mainly derived from [5], but as for all the other processes of the subsystems, stronly simplified. Essentially the process of fermentation consists of the sugar molecules which are already been broken down by the previous processes, are transformed into ethanol as a result of yeast cells. The paper referred to described 5 responses in the process of fermentation; biomass response, sugar level response, ethanol level response, diacetyl response and ethyl acetate. The 2 latter ones will not be discussed in this thesis, since they do not influence the response of the first 3, and are just an extra interest in the process of fermentation. The process of fermentation will not be taken into account in the updated scheduler, since it is presented as the last process and each of the jobs need to undergo this process tank. However, it is briefly outlined such that for further research a model for the fermentation can be used if for instance 2 fermentation tanks are available.

The biomass response, sugar level response and ethanol level response are briefly discussed, but also in these processes simplifications are made. As has been already mentioned, this is done because the aim of this thesis will not be the realistic behaviour of the subsystems, but rather their influence with variable processing times on the scheduler. In the referred paper the biomass is considered from the inoculum(the initial substrate) which is soluted into the wort. The inoculum consists of 3 type of yeast cells; Dead yeast cells, active yeast cells and lag cells. Normally in the process of fermentation, 2 phases are provided. In the first phase, just after the inoculum is soluted into the wort, the dead yeast cells boil down and get removed from the solution, as well as the lag yeast cells become active yeast cells with a specific rate μ_L. In the second phase the dead cells still boil down to get removed, the lag cells still become active, but in this phase the active cells take place. These active cells in the second phase can grow, or oppositely die. In this thesis only the second phase is taken into account. The amount of active cells is taken as a first state with the following dynamics. The lag cells are taken into account as well as a second state:

$$\begin{aligned} \tfrac{dX_{act}(t)}{dt} &= \mu_x X_{act}(t) - \mu_{DT} X_{act}(t) + \mu_L X_{lag}(t) \\ \tfrac{dX_{lag}(t)}{dt} &= -\mu_L X_{lag}(t) \end{aligned} \tag{4-73}$$

The yeast cells thereafter, will consume the sugar molecules to produce the ethanol in the beer. This is obtained by the following relation with the yeast cells:

$$\frac{dC_s(t)}{dt} = -\mu_s X_{act}(t) \tag{4-74}$$

And where the forming of ethanol with respect to the yeast cells is contained in the relation:

$$\frac{dC_e(t)}{dt} = \mu_e X_a ct(t) \tag{4-75}$$

Normally the rate in which ethanol is created is decreasing with time. This can be modelled by the addition of an inhibition term. Though, in this thesis, to let the system remain simple, the inhibition term is neglected. The terms μ_x, μ_{DT}, μ_L, μ_s and μ_e are described by the following relations. Some of the values are taken from the referred report, since they are obtained by experimental data.

$$\begin{aligned} \mu_x &= \tfrac{\mu_{x_0} C_s(t)}{k_x + C_e(t)}, \qquad & \mu_{DT} &= e^{(130.16 - \frac{38313}{T})} \\ \mu_L &= e^{(30.72 - \frac{9501.54}{T})}, \qquad & \mu_s &= \tfrac{\mu_{s_0} C_s(t)}{k_s + C_s(t)} \\ \mu_e &= \tfrac{\mu_{e_0} C_s(t)}{k_e + C_s(t)} \end{aligned} \tag{4-76}$$

The temperature is taken constant at $T = 283K$ such that the values for μ_{DT} and μ_L become constant. The values for μ_{x_0}, k_x, μ_{s_0}, k_s, μ_{e_0} and k_e are also constant with respect to the constant temperature:

$$\begin{aligned} \mu_{DT} &= 0.0054, \quad & \mu_L &= 0.0576 \\ \mu_{x_0} &= 0.108, \quad & k_x &= 1 \\ \mu_{s_0} &= 0.4783, \quad & k_s &= 1 \\ \mu_{e_0} &= 0.2988, \quad & k_e &= 3.4281 \end{aligned} \tag{4-77}$$

Now the state-space model can be obtained. By assigning the variables x_1, x_2 and x_3 and x_4 to the the states X_{act}, X_{lag}, C_s and C_e the model can be expressed in x as:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} \frac{0.108 x_3(t)}{1+x_4(t)} x_1(t) - 0.0054 x_1(t) + 0.0576 x_2(t) \\ -0.0576 x_2(t) \\ -\frac{0.4783 x_3(t)}{1+x_3(t)} x_1(t) \\ \frac{0.2988 x_3(t)}{3.4281+x_3(t)} x_1(t) \end{bmatrix} \tag{4-78}$$

Since dead yeast cells can be considered dead while they are actually lag cells, zero-mean white process noise $w(t)$ is added to x_2. For the rest of the model, it is assumed that it behaves without noise. Since yeast is not very nice measurable, the sugar levels are taken as measurements with zero-mean white measurements noise $v(k)$. This all can be elaborated in the following state equations:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} \frac{0.108 x_3(t)}{1+x_4(t)} x_1(t) - 0.0054 x_1(t) + 0.0576 x_2(t) \\ -0.0576 x_2(t) \\ -\frac{0.4783 x_3(t)}{1+x_3(t)} x_1(t) \\ \frac{0.2988 x_3(t)}{3.4281+x_3(t)} x_1(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} w(t) \tag{4-79}$$

Such that the discrete state space consists of:

$$\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+1) \\ x_3(\kappa+1) \\ x_4(\kappa+1) \end{bmatrix} = \begin{bmatrix} x_1(\kappa) + T_s(\frac{0.108x_3(\kappa)}{1+x_4(\kappa)}x_1(\kappa) - 0.0054x_1(\kappa) + 0.0576x_2(\kappa)) \\ x_2(\kappa) - T_s(0.0576x_2(\kappa)) \\ x_3(\kappa) - T_s(\frac{0.4783x_3(\kappa)}{1+x_3(\kappa)}x_1(\kappa)) \\ x_4(\kappa) + T_s(\frac{0.2988x_3(\kappa)}{3.4281+x_3(\kappa)}x_1(\kappa)) \end{bmatrix} + \begin{bmatrix} 0 \\ T_s \\ 0 \\ 0 \end{bmatrix} w(\kappa) \tag{4-80}$$

With the measurements as:

$$y(\kappa) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \\ x_3(\kappa) \\ x_4(\kappa) \end{bmatrix} + v(\kappa) \tag{4-81}$$

4-3 State estimation of the subsystems

In this paragraph the description of the estimations is done by the use of the state=space description from the previous section. Essentially, it is quite straightforward using the information about the extended Kalman Filter from chapter 3. The realizations of the estimations will be put into the results paragraph.

4-3-1 Mashing

As is already mentioned before, estimations need to be made from the measured data and system dynamics, such that processing times can be implemented beforehand. This will lead to a plausible better dynamical schedule. The discrete time state-space description can be formulated as:

$$\underbrace{\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+2) \\ x_3(\kappa+3) \end{bmatrix}}_{x(\kappa+1)} = \underbrace{\begin{bmatrix} x_1(\kappa) - T_s a_\alpha k_0 x_1(\kappa) - T_s a_\alpha x_3(\kappa) x_1(\kappa) \\ x_2(\kappa) + T_s a_\alpha k_0 x_1(\kappa) + T_s a_\alpha x_3(\kappa) x_1(\kappa) \\ \zeta x_3(\kappa) \end{bmatrix}}_{f(x(\kappa))} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix}}_{H(x(\kappa))} w(\kappa) \tag{4-82}$$

The data received from the measurements, can be described by:

$$y(\kappa) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \\ x_3(\kappa) \end{bmatrix}}_{x(\kappa)} + v(\kappa) \tag{4-83}$$

Where $w(\kappa)$ and $v(\kappa)$ are zero-mean white noise signals respectively with covariances $Q(\kappa)$ and $R(\kappa)$. For the extended Kalman filter algorithm described in the preliminaries, the following matrices can be obtained:

$$\frac{\partial f(\hat{x}(\kappa-1))}{\partial x(\kappa-1)} = \begin{bmatrix} 1 - T_s a_\alpha k_0 - T_s a_\alpha \hat{x}_3(\kappa-1) & 0 & -T_s a_\alpha \hat{x}_1(\kappa-1) \\ T_s a_\alpha k_0 + T_s a_\alpha \hat{x}_3(\kappa-1) & 1 & T_s a_\alpha \hat{x}_1(\kappa-1) \\ 0 & 0 & \zeta \end{bmatrix} \tag{4-84}$$

$$H(\hat{x}(\kappa-1|\kappa-1)) = H = \begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix} \tag{4-85}$$

$$\frac{\partial g(\hat{x}(\kappa|\kappa-1))}{x(\kappa)} = C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{4-86}$$

$$g(\hat{x}(\kappa|\kappa-1)) = C\hat{x}(\kappa|\kappa-1) \tag{4-87}$$

Filling these values into the extended Kalman filter algorithm described in the preliminaries preliminaries with initial conditions:

$$P(0|0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \hat{x}(0) = E[x_0] = \begin{bmatrix} 100 \\ 0 \\ 0 \end{bmatrix} \tag{4-88}$$

4-3-2 Brewing

For the brewing process, the same procedure is done as in the previous paragragh for the estimation of the mashing process. No further elaboration is necessary such that the estimations can be done by assigning the following equations.

$$\underbrace{\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+2) \\ x_3(\kappa+3) \end{bmatrix}}_{x(\kappa+1)} = \underbrace{\begin{bmatrix} x_1(\kappa) - T_s b_0 x_1(\kappa) - T_s x_3(\kappa) x_1(\kappa) \\ x_2(\kappa) + T_s b_0 x_1(\kappa) + T_s x_3(\kappa) x_1(\kappa) \\ \zeta x_3(\kappa) \end{bmatrix}}_{f(x(\kappa))} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix}}_{H(x(\kappa)} w(\kappa) \tag{4-89}$$

With the measurement equation:

$$y(\kappa) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \\ x_3(\kappa) \end{bmatrix}}_{x(\kappa)} + v(\kappa) \tag{4-90}$$

And with the rest of the necessary terms for the extended Kalman filter as:

$$\frac{\partial f(\hat{x}(\kappa-1))}{\partial x(\kappa-1)} = \begin{bmatrix} 1 - T_s b_0 - T_s \hat{x}_3(\kappa-1) & 0 & -T_s \hat{x}_1(\kappa-1) \\ T_s b_0 + T_s \hat{x}_3(\kappa-1) & 1 & T_s \hat{x}_1(\kappa-1) \\ 0 & 0 & \zeta \end{bmatrix} \tag{4-91}$$

$$H(\hat{x}(\kappa-1|\kappa-1)) = H = \begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix} \tag{4-92}$$

$$\frac{\partial g(\hat{x}(\kappa|\kappa-1))}{x(\kappa)} = C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{4-93}$$

$$g(\hat{x}(\kappa|\kappa-1)) = C\hat{x}(\kappa|\kappa-1) \tag{4-94}$$

Filling these values into the extended Kalman filter algorithm described in the preliminaries preliminaries with initial conditions:

$$P(0|0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \hat{x}(0) = E[x_0] = \begin{bmatrix} 100 \\ 0 \\ 0 \end{bmatrix} \tag{4-95}$$

4-3-3 Fermentation

The fermentation process has a slightly different behaviour than the 2 processes before. This is caused by the growth of the yeast which is the producer of the ethanol molecules. Lets consider the already obtained discrete time system:

$$\begin{bmatrix} x_1(\kappa+1) \\ x_2(\kappa+1) \\ x_3(\kappa+1) \\ x_4(\kappa+1) \end{bmatrix} = \begin{bmatrix} x_1(\kappa) + T_s(\frac{0.108x_3(\kappa)}{1+x_4(\kappa)}x_1(\kappa) - 0.0054x_1(\kappa) + 0.0576x_2(\kappa)) \\ x_2(\kappa) - T_s(0.0576x_2(\kappa)) \\ x_3(\kappa) - T_s(\frac{0.4783x_3(\kappa)}{1+x_3(\kappa)}x_1(\kappa)) \\ x_4(\kappa) + T_s(\frac{0.2988x_3(\kappa)}{3.4281+x_3(\kappa)}x_1(\kappa)) \end{bmatrix} + \begin{bmatrix} 0 \\ T_s \\ 0 \\ 0 \end{bmatrix} w(\kappa) \tag{4-96}$$

With the measurements as:

$$y(\kappa) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(\kappa) \\ x_2(\kappa) \\ x_3(\kappa) \\ x_4(\kappa) \end{bmatrix} + v(\kappa) \tag{4-97}$$

Now by continouing the same procedure one can obtain the matrices:

$$\frac{\partial f(\hat{x}(\kappa-1))}{\partial x(\kappa-1)} = \begin{bmatrix} (1-0.0054T_s) + T_s\frac{0.108\hat{x}_3(\kappa-1)}{1+\hat{x}_4(\kappa-1)} & 0.0576T_s & T_s\frac{0.108\hat{x}_1(\kappa-1)}{1+x_4(\hat{\kappa}-1)} & -T_s\frac{0.108\hat{x}_3(\kappa-1)\hat{x}_1(\kappa-1)}{1+\hat{x}_4(\kappa-1)^2} \\ 0 & 1-0.0576T_s & 0 & 0 \\ -T_s\frac{0.4783\hat{x}_3(\kappa-1)}{1+\hat{x}_3(\kappa-1)} & 0 & 1-T_s\frac{0.4783\hat{x}_1(\kappa-1)}{1+\hat{x}_3(\kappa-1)^2} & 0 \\ T_s\frac{0.2988\hat{x}_3(\kappa-1)}{3.4281+\hat{x}_3(\kappa-1)} & 0 & T_s\frac{0.2988\hat{x}_1(\kappa-1)}{3.4281+\hat{x}_3(\kappa-1)^2} & 1 \end{bmatrix} \tag{4-98}$$

And:

$$H(\hat{x}(\kappa-1|\kappa-1)) = H = \begin{bmatrix} 0 \\ T_s \\ 0 \\ 0 \end{bmatrix} \tag{4-99}$$

$$\frac{\partial g(\hat{x}(\kappa|\kappa-1))}{x(\kappa)} = C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \tag{4-100}$$

$$g(\hat{x}(\kappa|\kappa-1)) = C\hat{x}(\kappa|\kappa-1) \tag{4-101}$$

However, in this case study, since the fermentation process is the last process of the simple system illustrated in 4-1, all jobs need to go through the fermentation process. Since minimizing the cost function over $y(k+j) = x_5(k+j) + p_5(k+j)$ for $j = 0, ..., N_p$, it is essentially minimizing over $x(k+j)$ since there can nothing be controlled in the process time $p(k+j)$. For this reason, the minimization of the cost function does not depend on the variable process time in machine 5. For this reason, the whole process and variable end time of the fermentation tank is not taken into account, and the process is considered to have a constant process time of $p_5(k+j) = 1$ for $j = 0, ..., N_p$. The elaboration of the state-space, as well as the elaboration of the use of the extended Kalman filter, are mainly proposed for future

research in which there are multiple fermentation tanks, and therefore multiple outputs of the batches. Also note the chosen value for the process noise is arbitrarly, this can be changed into a parameter varying additional state as is done for the mashing and brewing. However, in this case study this additional features are not discussed.

Chapter 5

Case study - Results

In this chapter, the methods discussed in the previous chapter are realized with determined parameters. The processing times per machine are assigned, and the translation from the process times of the machines to the state-space descriptions of the subsystems are outlined. Furthermore, estimations are obtained, in which the predictions subsequently follow. Reliability of the predictions is discussed as well, where afterwards the results for the statical scheduler, the dynamical updated scheduler with respect to the event-counter k, and the dynamical updated scheduler with respect to the time-counter κ are shown.

5-1 General assumptions and explanations

For the realizations of the subsystems, as well as to conclude anything about the whole model, several assumptions need to be made, and explanations need to be given beforehand. These assumptions can lead to the system not behaving according to reality, but since this is a fictitious case study and the focus is lied more upon the behaviour of the scheduler with updating processing times rather than the subsystems behave like realistic subsystems, this is taken for granted. The first assumption which is made regarding the subsystems, has already been mentioned in previous paragraphs. Namely, the last process of fermentation, will be neglected with respect to its model dynamics. This is done because in our case study, only one fermentation tank is included, where all jobs need to go through this specific fermentation tank. Minimizing the overal cost over the endtimes of the jobs is essentially the same as minimizing over the starting times of this last machine. Deviating processing times of this machine will therefore not influence the mode switching. In extension, too long processing time of this last machine will lead to a loss of beneficial mode changes in previous machines. To conclude, a low and constant value for the process time of machine 5 is chosen $p_5 = 1$. The other processing times are summarized in table 5-1.

The values of processing times of the other processes are taken such that the processing times of the first 2 machines for mashing, have a higher overal processing times than for the

Machine	Process time(Days or "Time-units")
M_1	7
M_2	8
M_3	6
M_4	5
M_5	1

Table 5-1: Nominal process times of the processes per machine.

second pair of machines for brewing. This is done such that at the beginning of the jobs, no machine would become idle, due to the machines 3 and 4 are taking too long. In other words, optimality is hard to conclude when machines 3 and 4 are working constantly due to longer processing times than machines 1 and 2. Therefore, the opposite is chosen, machine 1 and 2 take longer, such that deviations in the processing times of machine 1 and 2, could result in fluctuations of routes in machine 3 and 4. Furthermore, the same processes deviate only 1 time-unit from one another to observe more clearly when mode changes are logical. In other words, when processing times deviate, it does not take too much to deviate to change the most optimal path in the perspective of one job, or the consecutive job.

Above all, comparisons need to be done to conclude anything about the beneficial effects of the updated scheduler. This is rather harder than it seems to be. Namely, processing times will deviate, but how to differ the benefits or delays from the realizations of the subsystem its process times instead of the updated scheduler actually being better or worse. To do this, the sum of differences with the nominal processing times is calculated for both the initial scheduler and the ultimate updated scheduler at the end of the process. By observing the processes who became "active" according to the modes of both the initial scheduler and ultimate updated scheduler, the sum can be obtained of the total differences with the nominal processing times for both the schedulers. Finally the differences are compared, and extracted from one another to obtain the difference from the schedulers with respect to each other. This is than substracted or added to the cost function of the initial scheduler such that comparison can be done. This is elaborated a lot more in the results paragraph which are coming up by the tables where the active processes are given in bold for both the schedulers. It could occur, the initial scheduler becomes active in a machine or process, where no realization is obtained for. For instance, when the initial scheduler for job 3 goes through machine 4, but the updated scheduler decided job 3 goes through machine 3 instead of machine 4, no realization is obtained for job 3 initially going through machine 4, since this process never happened. Just come up with another realization would make the comparison even more dubious, since the overal comparison becomes a comparison between sort of random realizations. Therefore, if the processes for the initial scheduler do not become active, it is assumed they have the same process time as the nominal process time, such that the difference with the nominal process times also equals 0.

To continue with the above information, another problem pops up. Namely, it could occur, the system changes modes from information about the processes of specific machine in machine 1 and 2, therefore switch the routes. However, it subsequently occurs the switching choice was a bad decision, due to the fact the realization in the machine 3 and 4, seem to be

very unreliable. In other words, when the nominal times would have occured in machines 3 and 4, it would be the best mode switching, but the realizations turn out to be very non-beneficial for the updated scheduler. A sort of random realization, followed up by another random realization could therefore become very hard to optimize over. Although the realizations are not completely random of course, and information is known beforehand, this is still a possible outcome. To conclude anything about this, in the following paragraphs about the results of the updated scheduler, 3 approaches are used for each of the dynamical schedulers obtianed. In the first approach, only the first 2 machines will be subjected to deviating processing times, in the second approach only the third and fourth machines will be subjected to deviating processing times, and in the last approach all machines except for machine 5, will be subjected to deviating processing times. Hopefully, something can be concluded about the above concerns.

Ultimately, for the comparison to be complete, it is assumed the input time-instances are all equal to 0, i.e. $u(k) = 0$ for $\forall k$. This is already mentioned in previous paragraphs as well. This is done, such that a beneficial effect of the scheduler can not be ruined by an input time-instant being too late. Every beneficial effect can subsequently be concluded to the effect of the updated scheduler.

5-2 Subsystems

5-2-1 Simulations

In this paragraph, the translation from the assigned processing times to the specific processes is discussed. Essentially, it comes down to tuning parameters in the state-space model such that the noiseless case behaves exactly as the prescribed processing times from table 5-1.

To begin, and actually also to explain, we take the first mashing tank, machine 1, as an example. The elaboration for the other processes will follow quite straightforward. By implementing the state-space description of 4-66 without additive noise in MATLAB, and constructing a simulation for $x_2(\kappa)$ of the state-space model with step size $T_s = 0.01$, simulation time $\kappa = 20$ days and distinct values for the parameter k_0, figure 5-1 is obtained. The state $x_2(\kappa)$ represents the amount of smaller sugar molecules made from the larger sugar molecules. The initial state for both the mashing tanks is $x(0) = \begin{bmatrix} 100 & 0 & 0 \end{bmatrix}^T$. The initial states $x_1(\kappa)$ and $x_2(\kappa)$ can also be considered as starting amounts of the molecules, but since they are converted in one another, it can be seen as a percentage more or less, such that $x_1(0)$ is 100 means it starts with 100% of its amount. $x_2(0) = 0$ means from the 100% of $x_1(0)$ still 0% is converted into the smaller sugar molecules. More or less, since the measurement noise can influence the data a bit, such that they are not fully converted in one another, The x-axis is counted in mDays or actually m('Time-units'), such that the end time of 20000 'm(Time-units)', means $\kappa = 20$ in the overal model. The goal is to design a certain threshold where the process is considered to be finished. As can be observed, the behaviour of the process is kind of asymptotically to the amount of 100. Taking a large value for k_0 brings the system faster to the asymptotic line of 100%, taking a smaller value for k_0 brings the system slower to the asymptotic value of 100%.

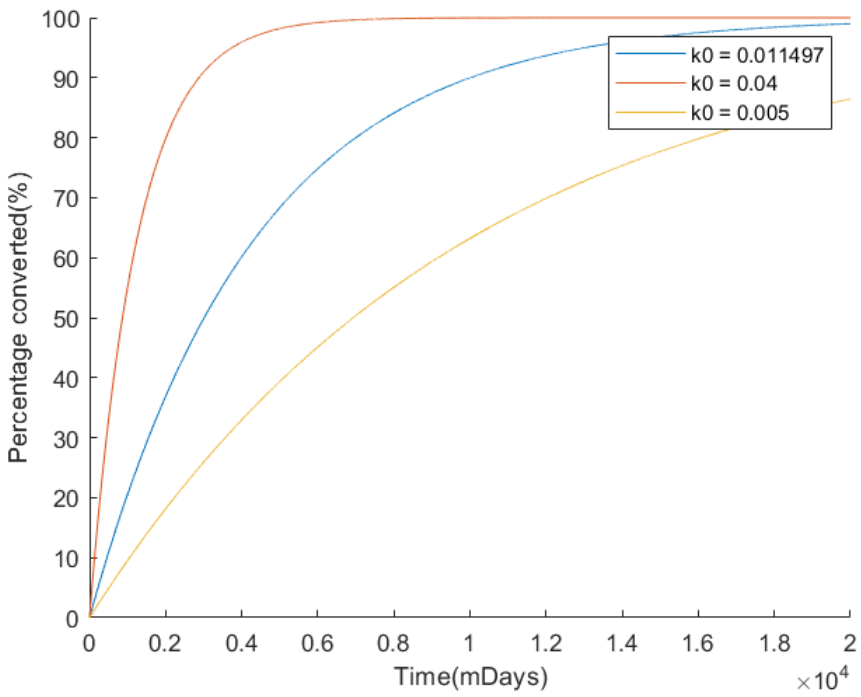

Figure 5-1: Simulation of $x_2(\kappa)$ in the mashing process with no noise and different values k_0.

Since the parameter k_0 in the real model is subjected to noise variables, and it can already be observed in figure 5-1 that small deviations in the value of k_0 already lead to a large difference taking 100% as a threshold, a more reliable threshold is chosen. Note that the k_0 values in figure 5-1 are taking constantly different, i.e. $k_0(\kappa) = k_0$ for $\forall\kappa$, where in the case noise is applied, $k_0(\kappa)$ will vary around the value of the initial k_0, but changes over time. Still though, another threshold than 100% need to be chosen such that the deviations in the time the threshold is reached do not deviate thoroughly. For that reason, 80% as a more reasonable threshold is chosen. In this threshold still the non-linearity of the system its dynamics is taken into account(the system behaves more or less linear when having the threshold closer to 0%), but also the times for reaching the threshold do not deviate too extreme when the value of $k_0(k)$ deviates.

By taking the threshold as 80%, the job is now to tune the parameter such that the threshold without noise is exactly met at the processing time from table 5-1. For instance, if we take the processing time of the first mashing tank $p_1 = 7$, the goal is to design k_0 such that with the 'no-noise' case it comes approximately at 80% at $\kappa = 7000$mDays which is of course, since the axis is in 'milli' equal to $\kappa = 7$. For the mashing tank 1, this is reached at the value of $k_0 = 0.011497$, as is already illustrated in figure 5-1. For the other processes, the same method can be applied quite straightforward to come up with appropriate values of k_0 or in case of the brewing b_0.

Now the process times of the initial scheduler are translated to the nominal trajectories of the state-space models of the processes by tuning the parameter k_0 or b_0. Next the process noise

need to be applied to make the system behave slightly different. The process noise is considered to be zero-mean white noise with covariance Q. An additional interest could be to apply non-gaussian or coloured noise, but this will not be in the scope of this thesis. The results are shown in figure 5-2. It can be observed that a too high covariance of the process noise,

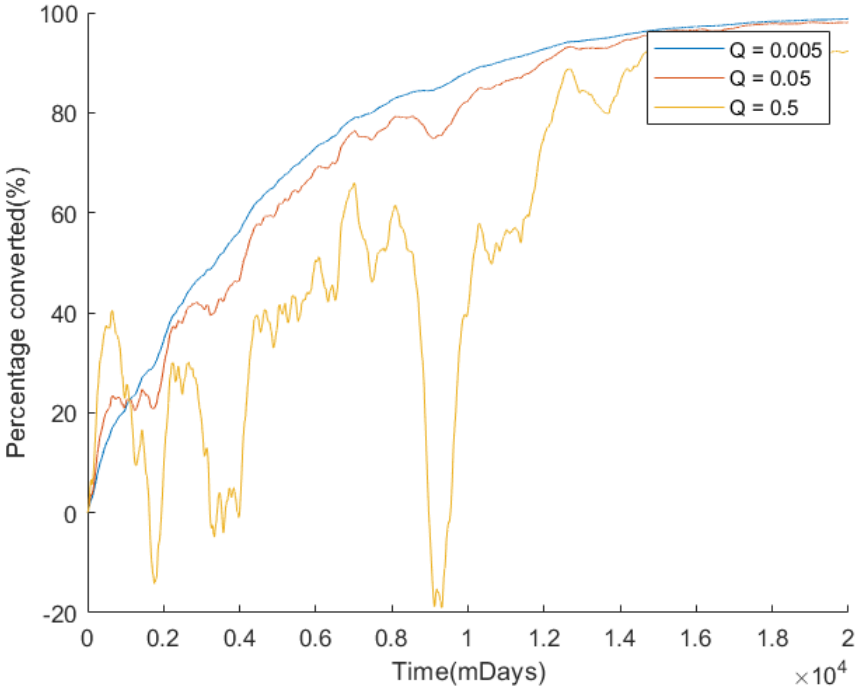

Figure 5-2: Simulation of $x_2(\kappa)$ in the mashing process with constant $k_0 = 0.011497$, $\zeta = 0.99$ and different covariance values for Q.

will not only let the system deviate from the simulation with no noise in figure 5-1, but could also completely overrule the system dynamics when for example looking at the trajectory for $Q = 0.5$. Note that due to the too high noise sequence, also solution come up which are infeasible looking at the negative percentages of the sugars converted. Also note that for the 3 realizations with different values for the covariance, the same seed is used for the sequences of white noise signals. Thereafter, they are multiplied by the square root of the covariance, such that the same noise sequences can be compared with different covariances. Besides the fact the system its behaviour is completely overruled when taking the value of the covariance Q too high, also the times the trajectories arrive at the threshold is significantly different. In this particular cases the time-instances the threshold of 80% is reached are shown in table 5-2. Of course the latter case in which $Q = 0.5$ is useless for the simulation since the model totally not behaves as the dynamics. Therefore an appropriate value of Q is wanted such that the system could deviate from the nominal trajectory without noise, but still follows the dynamics sufficiently well. As can be seen, the value of $\zeta = 0.99$ is taken constant. Though, this parameter can also be tuned. To conclude anything about the influence of this parameter ζ, also called the memory factor, the following situations with 'worse case scenario' of $Q = 0.5$ is picked to illustrate. What can be clearly observed in figure 5-3 is that the memory factor blushes out the strong covariance of the noise. Of course this is logical when looking at the

Q	Time(mDays)
0.005	7481
0.05	9703
0.5	12241

Table 5-2: Threshold(>80%) arrival times with distinct covariance Q.

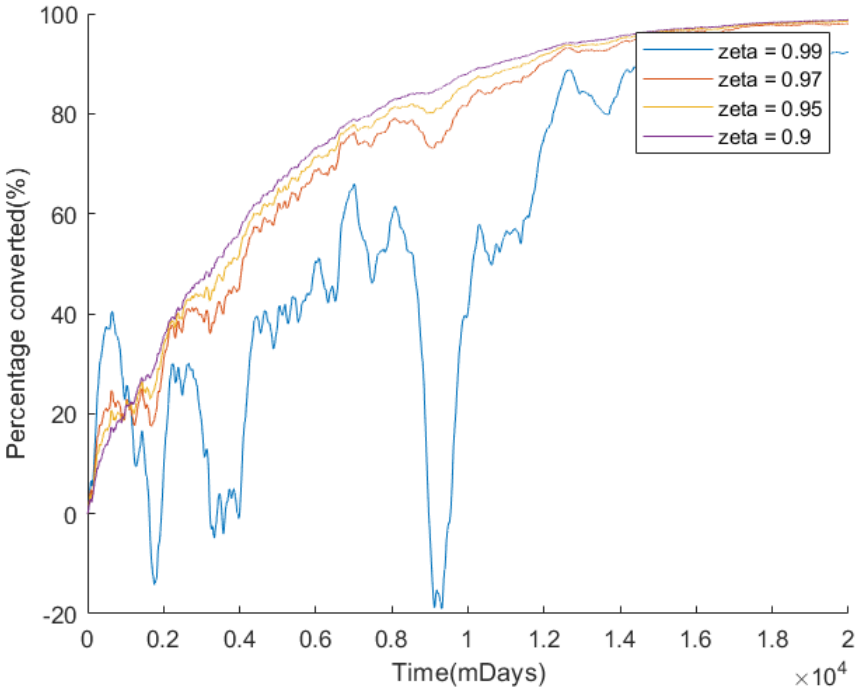

Figure 5-3: Simulation of $x_2(\kappa)$ in the mashing process with constant $k_0 = 0.011497$, $Q = 0.5$ and different values for ζ.

state-space equation for $x_3(\kappa)$ in 4-66. Since the dynamics evolve according to:

$$x_3(\kappa + 1) = \zeta x_3(\kappa) + w(\kappa) \tag{5-1}$$

When taking ζ closer to 0, the influence of the noise term becomes less and less important, since it is canceled out by the larger decrease to zero because of the ζ-term. Constructing the times where the trajectories of 5-3 reach the threshold of 80%, the table 5-3 is obtained. Of course, this results in the processing time for the specific realizations becoming closer to the nominal processing time. The trade-off which need to be made is, in which way we want to simulate the influence of the covariance of the process noise, with the influence of the memory factor ζ, with as a goal it deviates from the prescribed processing times, but can still be predicted quite well. Therefore, the trajectories need to be most familiar with the nominal trajectory, to make accurate predictions, but deviations with respect to the nominal time. In the section about predictions, this is much more elaborated.

To come up with the appropriate values for k_0/b_0 and ζ for the simulation of the subsys-

ζ	Time(mDays)
0.99	12241
0.97	9702
0.95	7889
0.90	7460

Table 5-3: Threshold(>80%) arrival times with distinct memory factor ζ.

tems, one more condition need to be added. Namely, the values of the k_0/b_0 can physically never be less than zero. If we look at the case where $Q = 0.05$. The values for the parameter with noise are equal to $k_0 + x_3(\kappa)$, as they can be observed in figure 5-4. It can be clearly

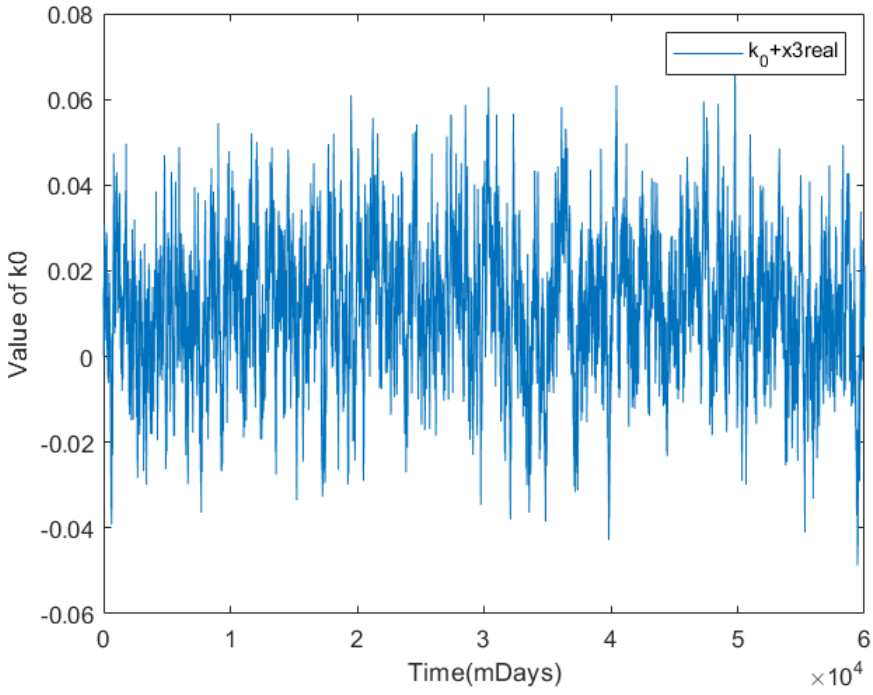

Figure 5-4: Simulation of $k_0 + x_3(\kappa)$ for $Q = 0.05$ and $\zeta = 0.99$.

noticed that the values of k_0 are often lower than 0, which is physically not feasible. To come up with a boundary of the covariance such that k_0 will always have realistic values, we take the covariance such that its standard deviation multiplied by 3 is never greater than the value of k_0, such that the probability is practically 0 that one of the values of $w(\kappa)$ is larger than k_0. There is still a very small probability that it occurs one of the values could become larger than k_0, but in this model we take that for granted, since the influence will be negligible.

The solution becomes to have the largest possible value for the covariance Q, such that k_0 remains more or less realistic and feasible, but thereafter tune the memory factor ζ such that the system still deviates in processing time, but is still quite nicely predictable. For the

mashing tank 1, this is done by setting the covariance to:

$$\sigma = \frac{k_0}{3}, \quad \sigma^2 = Q \tag{5-2}$$

Tuning the variable ζ the same as in figure 5-3, gives the value for $\zeta = 0.9999$. To give an insight, 100 realizations are obtained ofr mashing tank 1 and put in figure 5-5. As can

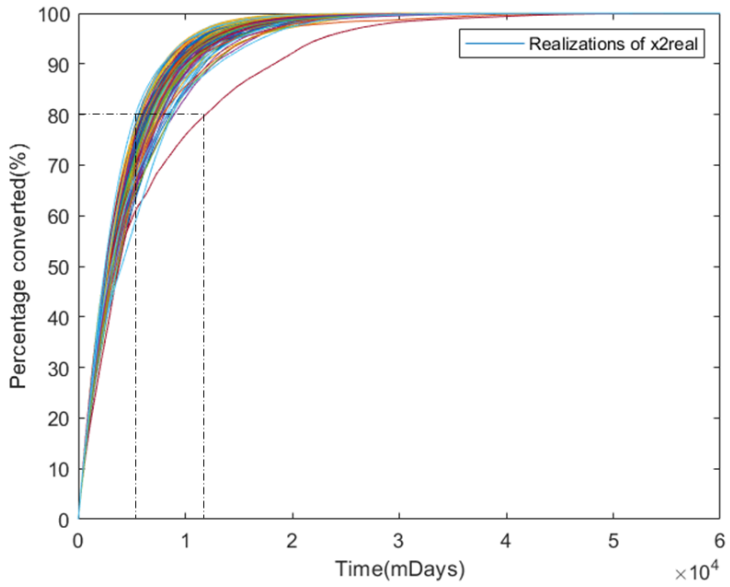

Figure 5-5: Monte-carlosimulations with $n = 100$ for state $x_2(\kappa)$of mashing tank 1 with $Q = \frac{k_0}{3}$ and $\zeta = 0.9999$. The dotted lines are the most extreme realizations where the threshold of 80% is reached.

be observed, the realizations have a very nice clear behaviour in a smooth line without disruptions, but deviate in processing time with respect to the threshold value of 80%. For the sharp eye, the process times to reach the threshold $> 80\%$ deviate from approximately $\kappa = 5000 - 9000$mDays, read $\kappa = 5 - 9$ with one outlier to approximately $\kappa = 12000 mDays$, read $\kappa = 12$. The same procedure is continued for the processes of the second mashing tank, and both the brewing tanks. For now, to conclude this paragraph, the parameters with respect to processing times for the process of mashing and brewing are summarized in table 5-4. Of course, the question arises if the behaviour of the brewing dynamics is not much different

Machine	Processing times	k_0/b_0	Q	ζ
M_1	7	0.011497	$1.469{\cdot}10^{-5}$	0.99990
M_2	8	0.01006	$8.884{\cdot}10^{-6}$	0.99970
M_3	6	0.026829	$7.998{\cdot}10^{-5}$	0.99991
M_4	5	0.032194	$1.152{\cdot}10^{-4}$	0.99992
M_5	1	-	-	-

Table 5-4: Parameters for the mashing and brewing process with distinct processing times per machine.

than the one from the mashing and therefore the behavioural aspects of the dynamics with respect to the noise and memory factor is different. Fortunately, the behaviour is more or less the same, such that the same procedure can be done as described for the mashing tank above. As is already mentioned at the beginning of this chapter, the fermentation process is considered to be constant, due to the fact minimizing the fastest outcome of the product of machine 5, is the same as minimizing the starting time of machine 5.

Since the measurements do not influence the real obtained data, the simulations of the measurements are not taken into account in this section. In the next section, realizations are made for the measurements, since they are primarily needed for the estimations. The choice for the covariance of the measurement noise R is also substantiated in the next section.

5-2-2 Estimations

In the previous section, the simulations of the subsystem are explained such that 'real' simulated data is gathered. In this section, the results are shown for the estimation for the specific states using the system dynamics and measurements from the obtained simulated data. The measurements are also contained with measurement noise. An extended Kalman filter is used as is already explained in the previous chapter.

For each of the processes, the results are shown with respect to the difference of the real process and the estimation. Especially important is the threshold point, where the estimated state must be very close to the real state, such that if we know the estimations are passed the threshold, the real systems will be more or less passed the threshold as well.

For the estimations, it is not only important what the process noise does to the system, but also how pure the measurements are. In other words, what influence the measurement noise contributes to the pureness of the estimations. In this paragraph, the same conclusions can be drawn from 1 mashing tank to the other mashing and brewing tanks with respect to the quality of the estimations, as the state-spaces of the mashing and brewing tank behave, within a certain degree, the same. In chapter 3, the method for the extended Kalman filter is discussed, where in the previous chapter the matrices are already obtained. Now by just filling the values into the algorithm and updating per prediction and correction step receiving the data $y(\kappa)$ from each of the subsystems, the estimations are derived.

First a general result is given from all the 3 states for mashing tank 1 when applying a certain process noise and measurement noise over the system. To begin, a deviation is made regarding the process noise where 3 cases are proposed, taken into account equation 5-2 and taking a seed such that the noise sequences behave the same. For all three cases the value $\zeta = 0.99$ is chosen, and the measurement noise covariance has the value $R = 0.1$. For the process noise:

$$Q = 0.005, \quad Q = 0.05 \quad Q = 0.5 \tag{5-3}$$

In the next figures respectively the estimations and real data of the simulations of the 3 states of the mashing process are shown. For all the 3 values of the covariance of the process noise, the estimations are working very well. The trajectories are almost perfectly aligned with the

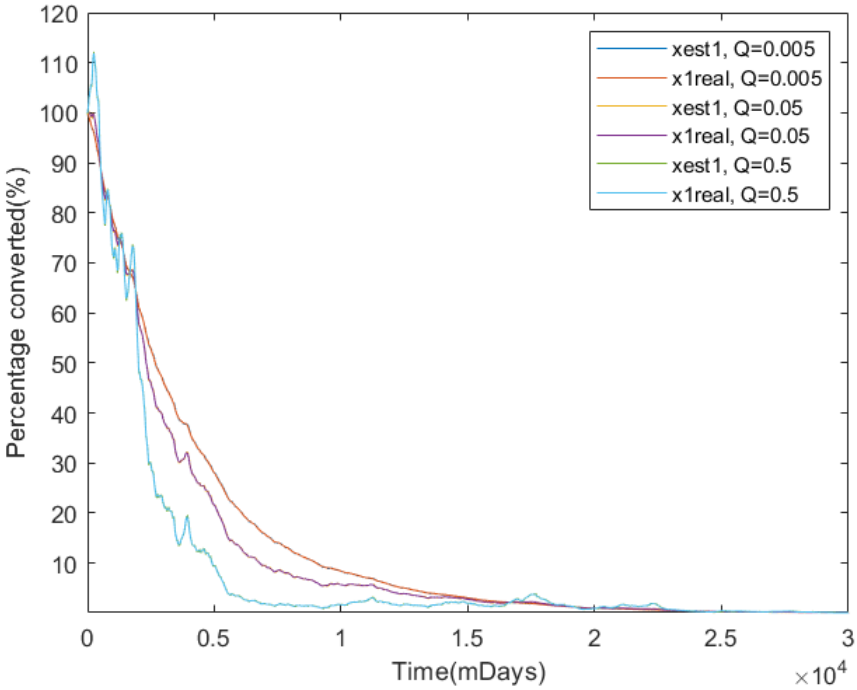

Figure 5-6: Estimations compared with the real data of the state $x_1(\kappa)$ with deviating process noise covariance Q.

trajectories of the real data such that they are barely recognizable in the figures. Note that the time-axis of each of the three figures 5-6, 5-7 and 5-8 are different. Now looking at the last state $x_3(\kappa)$ containing all the noise in figure 5-8, it can be observed that the estimations for the 3 cases work quite well, but get a larger off-set when the time increases. A possible explanation for this behaviour is that the system its response from the process noise encountered is negligible when the time increases. This is caused by the state $x_1(\kappa)$ becoming almost or approximately 0. Therefore the term in equation 4-66, $(T_s a_\alpha x_3(\kappa) x_1(\kappa))$, also becomes 0. By taking only the measurements from $x_1(k)$ it becomes impossible to estimate $x_3(\kappa)$ from the term $(T_s a_\alpha x_3(\kappa) x_1(\kappa))$, since this last term is 0 and the influence of $x_3(\kappa)$ to this term is therefore nihil. Therefore taking the process noise either 0, or some other value close to 0, will not give a deviation in the measurements such that the process noise in $x_3(\kappa)$ can be certified.

In the above conclusion about the inability of estimating the state $x_3(\kappa)$ when time increases, a link can be made to one of the key features for the estimations, the measurements. The measurements are only taken from state $x_1(\kappa)$ in the defined state-spaces. Of interest is of course to see the response and estimation of the extended Kalman filter when making the measurements less pure. In other words, apply measurement noise with a higher covariance R. Taking the value of the process noise constant at the largest noise case $Q = 0.5$, the measurement noise is deviated with the following values:

$$R = 0.1, \quad R = 1, \quad R = 10; \tag{5-4}$$

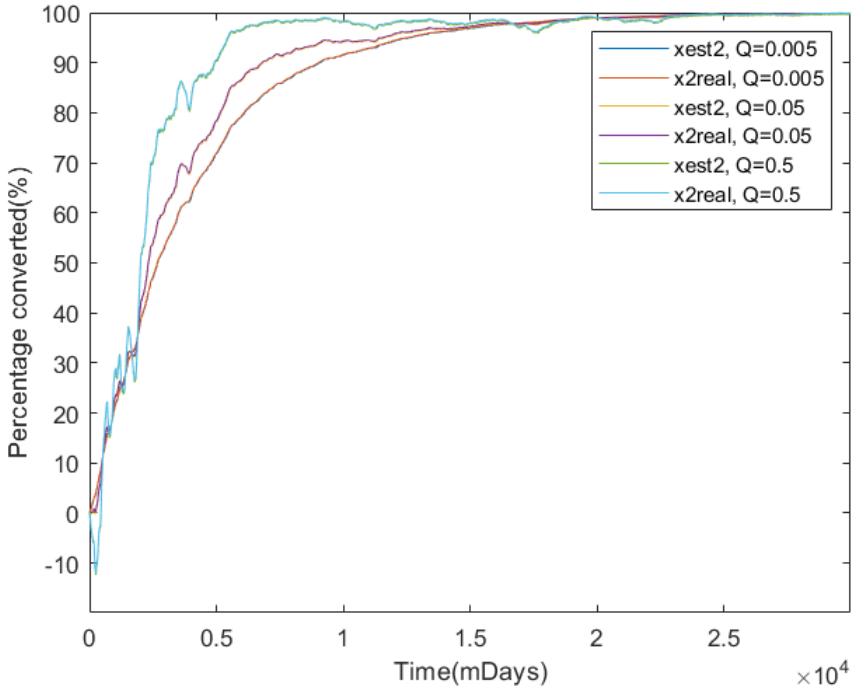

Figure 5-7: Estimations compared with the real data of the state $x_2(\kappa)$ with deviating process noise covariance Q.

The next figures show the estimation of the threshold state $x_2(\kappa)$ and the measurements $y(\kappa)$ of the three specific covariance for the measurement noise. The figures are shown in figures 5-9 and 5-10. As always for the comparison, the noise sequences are seeded. Obviously the measurements become very noisy with an increasing covariance R, as can be observed in figure 5-9. However, the estimations in figure 5-10 are not influenced by it that much, and still follow the real simulated data quite nice. Of course if you zoom in, the differences become significant, but the overal estimations still follow the real simulated data remarkably nice.

Thereafter, it is nice to observe what the system does with the assigned values of the covariance of the process noise and memory factor to the subsystems described in the previous section about simulations. The same procedure is done, only this time the difference from the real obtained datapoint for the threshold >80% is compared. Since the estimations in a figure are very hard to tell apart, the difference of the estimation with respect to the threshold point is given in table 5-5. As can be concluded, the value of the threshold point reached for

	R = 0.1	R = 1	R=10
$x_2(\kappa)_{\text{real}}$	7051	7051	7051
$x_2(\kappa)_{\text{est}}$	7053	7069	7110

Table 5-5: Comparison of the theshold point >80% reached in mDays for the real simulated data and the estimations.

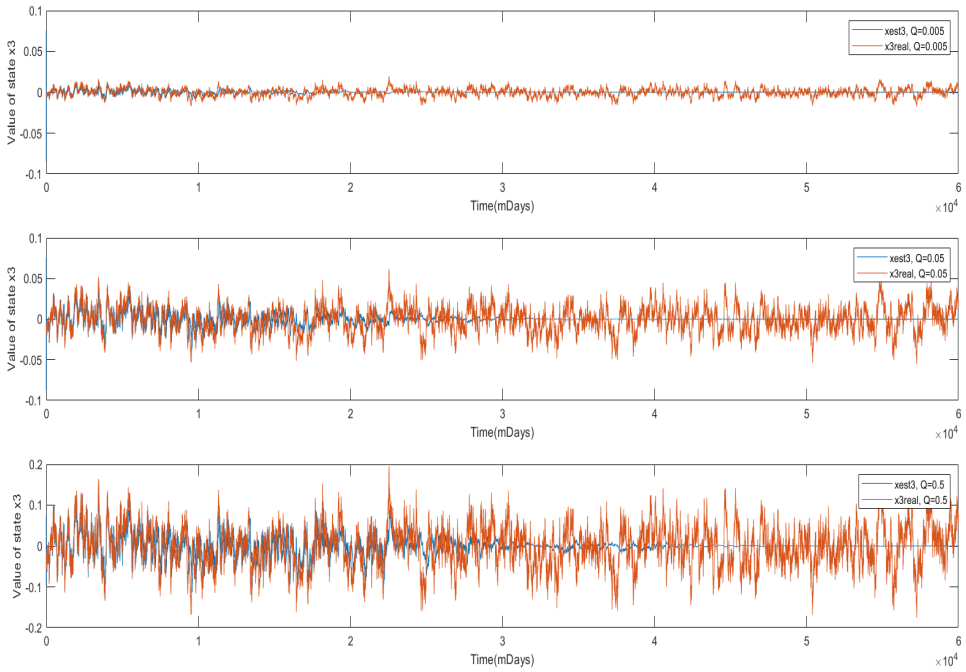

Figure 5-8: Estimations of the state $x_3(\kappa)$ compared with the real data with deviating process noise covariance Q. In consecutive order from $Q = 0.005$, to the second image $Q = 0.05$, to the third image $Q = 0.5$.

the estimations floats away as the measurements become less pure. Of course, that is logical, since the estimations of the processes become less accurate when the measurements become more noisy, as is already shown in figure 5-10. Since the estimated threshold point are the only data known in a realistic situation, the measurement noise is chosen to be very small, such that the sensors are very reliable. A more commonly used value for the covariance of the measurement noise is $R = 0.1$ as this value will also be used in this thesis for the covariance of the measurement noise.

To ultimately finish the estimation process, one more thing has to be noticed. For the prediction in the next section, accurate estimation are needed for all the 3 states. For the first 2 states, the estimations are very nice, for the third state however, estimations become less and less accurate when the measurements are too noisy. The estimation still follow the trend of the real simulated value of $x_3(\kappa)$ up to a certain time. The off-set can also be seen when time increases and is prabably caused by the reason already mentioned above. Namely, the value of $x_1(\kappa)$ coming closer and closer to 0, makes it harder to estimate the correct value since the influence of the state $x_3(\kappa)$ to the measurable state $x_1(\kappa)$ becomes lesser. However, even in the beginning, the estimations are not so pure. Taking almost perfect measurements such that the value of $R = 0.0001$, the figure 5-12 is obtained. It can be observed that the estimation in figure 5-12 is much better and has a much lesser off-set than the estimation in figure 5-11, also when time increases. The point to be made by showing these 2 figures, is the reliability of the predictions in the next section, with the chosen value for R as the covariance

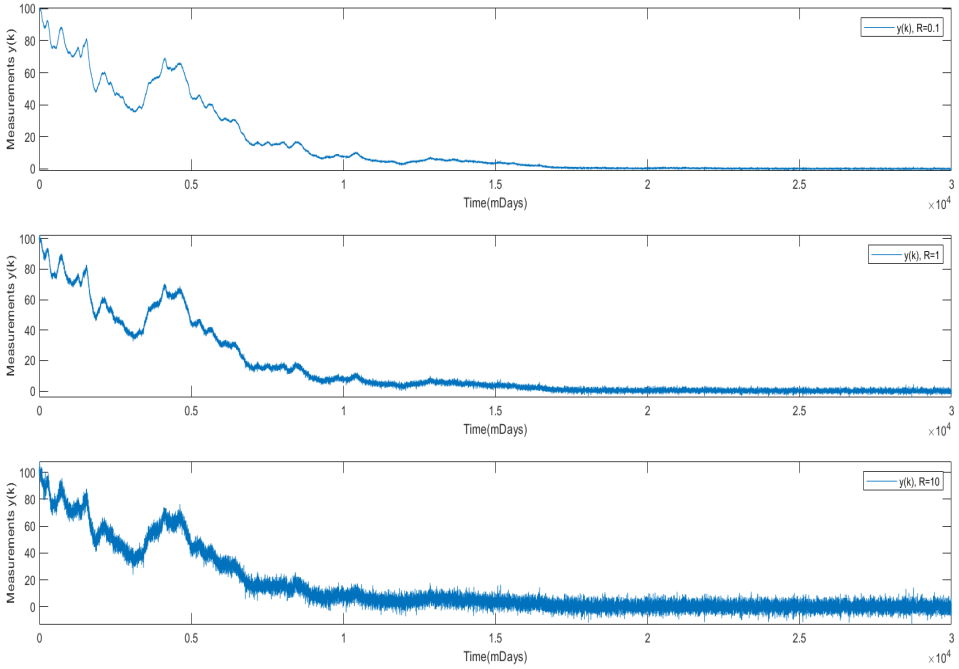

Figure 5-9: Influence of increasing covariance of the measurement noise R on the measurements $y(\kappa)$ of $x_1(\kappa)$ with covariance of the process noise as $Q = 0.5$. The first image $R = 0.1$, the second image $R = 1$ and the third image $R = 10$.

of the measurement noise. The purer the measurements, i.e. the smaller value for R, the better all the states including $x_3(\kappa)$ get better estimates. As a result, the predictions with these better estimated states become more reliable. The question which will be answered in the next section is how reliable the predictions are when taking the values for the covariance of the process and measurement noise as in table 5-4 and $R = 0.1$.

5-2-3 Predictions

In the previous chapter is explained that the estimations do not deviate too much from the real system, and thus the estimated threshold point can be considered as the real threshold point. Of course, there is no other choice, since the only data we can obtain from $x_2(\kappa)$ is from the virtual sensor of $x_2(\kappa)$, i.e. the estimated values of $x_2(\kappa)$. In this section, the results for the predictions of some realizations are given to make conclusions about the reliability and behaviour of the predictions.

The problem encountered will be if the predictions are reliable enough, and more specific, convert to the real end time of the process. In other words, the scheduler can not do anything with a prediction which is chattering around the prescribed process end-time. For instance, when having mashing tank 1, with a process time of $p_1 = 7$, predictions are not very useful if they are chattering largely around the value of $\kappa = 7$. If the predicted value at $\kappa = 3$ is $p_1 = 8.5$ but a moment later at $\kappa = 4$ the predicted value becomes $p_1 = 6$, the scheduler can

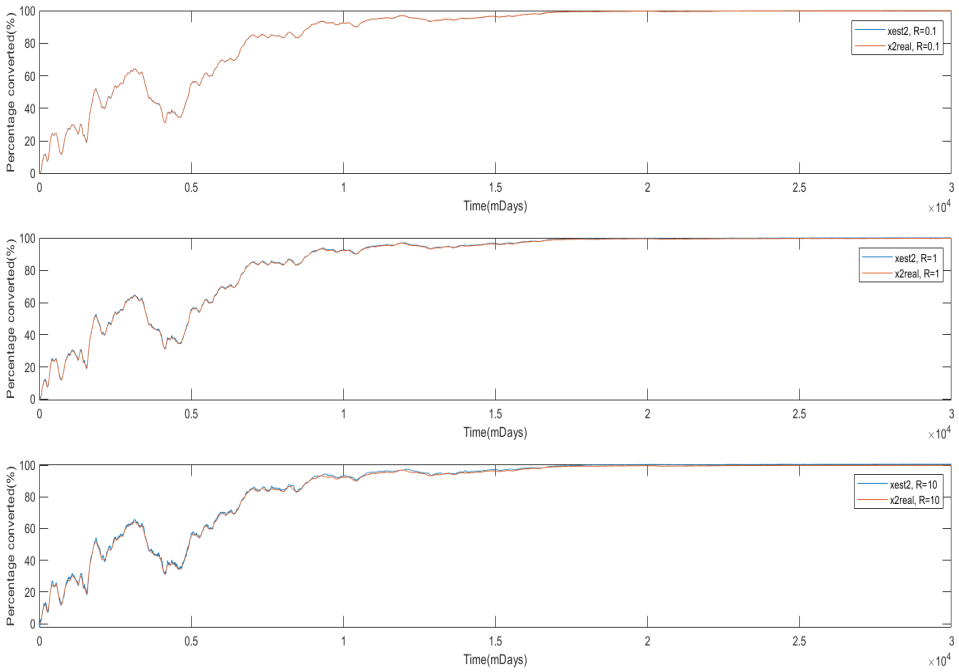

Figure 5-10: Influence of increasing covariance of the measurement noise R on the estimations of $x_2(\kappa)$ with covariance of the process noise as $Q = 0.5$. The first image $R = 0.1$, the second image $R = 1$ and the third image $R = 10$.

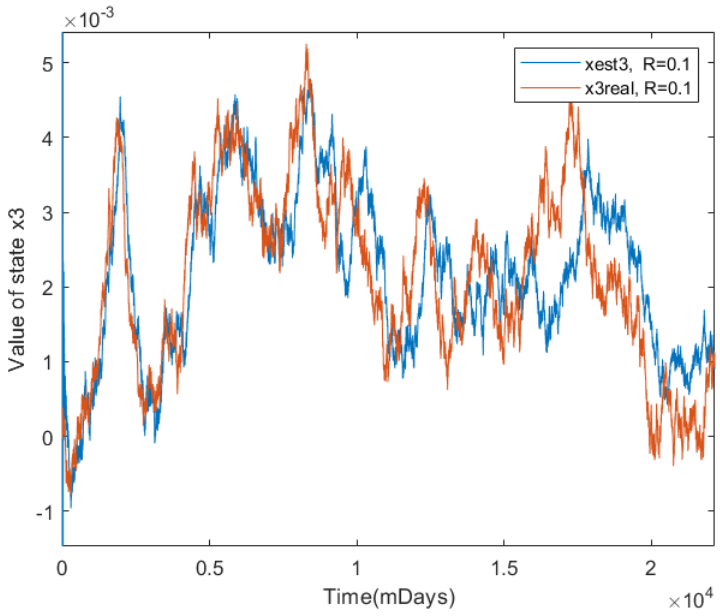

Figure 5-11: Comparison of the estimation and real simulated data of $x_3(\kappa)$. The real model parameters are used with covariance of the measurement noise as $R = 0.1$.

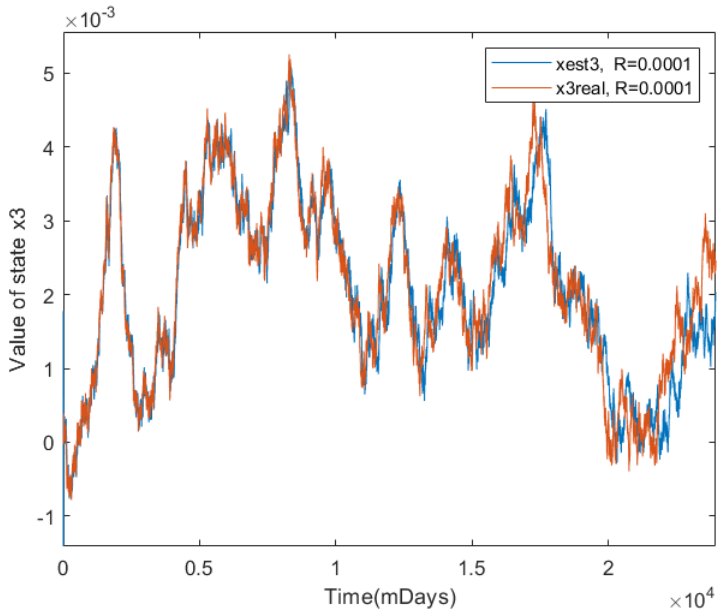

Figure 5-12: Comparison of the estimation and real simulated data of $x_3(\kappa)$. The real model parameters are used with covariance of the measurement noise as $R = 0.0001$.

not make a useful logical update. A preferable situation will be if the predictions become more certain when time increases, and convert to the endtime.

First the results of the general approach are shown. Essentially, it comes down to estimating the 3 states of the subsystems, and continue filling in the value in the state space of this estimated 3 states without the noise. Or, with the expectation of the noise, but since the noise is considered to be zero-mean white noise, the expectation equals 0 such that the noise can be left out of the state-space when predicting the system. Hopefully, predicting the states at $\kappa = 1$, gives a less accurate prediction, than predicting the states at $\kappa = 4$ for instance, since less of the estimated data is known at the earlier time-instances. In figures 5-13 and 5-14, the approach is nicely illustrated. First of all, it can be concluded again, the estimations work almost perfectly since the red and the green line are perfectly aligned on each other. The red line is considered as the estimations of the real data up till the time the data is known. Afterwards, the prediction in blue 'takes over' the red line. For the sharp eye it can be observed that the blue line comes closer to the green line when time increases comparing the figures 5-13 and 5-14. Specifically looking at the threshold point of >80%, it also becomes closer. Since it is a little bit hard to observe if from the graph, the next table 5-6 will show the results in a more organized way. The behaviour in table 5-6 is a nice

	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$	$\kappa = 4$	$\kappa = 5$	$\kappa = 6$	$\kappa = 7$	$\kappa = 8$
Prediction reaches threshold point	6.80	5.69	6.56	6.33	6.11	6.01	6.01	6.01
Real data reaches threshold point	6.01	6.01	6.01	6.01	6.01	6.01	6.01	6.01

Table 5-6: Comparison of the predicted threshold time reached with the real threshold time reached for increasing κ.

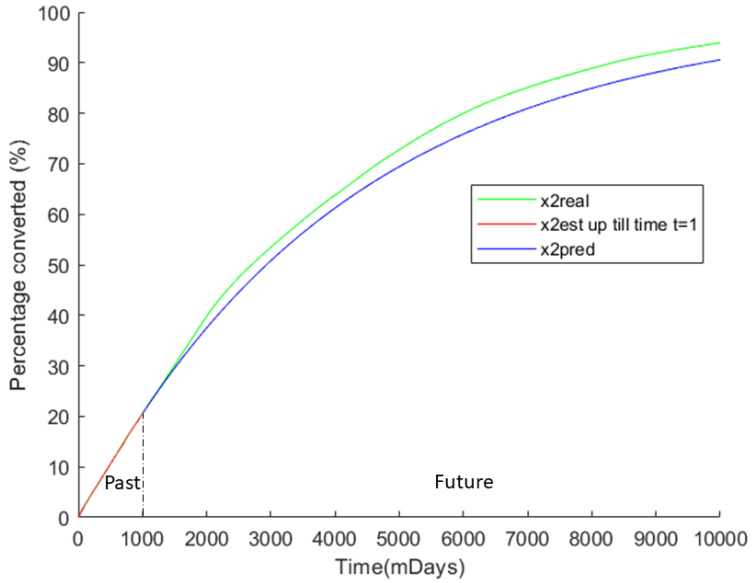

Figure 5-13: Prediction at time $\kappa = 1(\kappa = 1000$mDays) for mashing tank 1. In red the estimation up till time $\kappa = 1$, in green the real simulated data of the system, in blue the prediction upward from $\kappa = 1$.

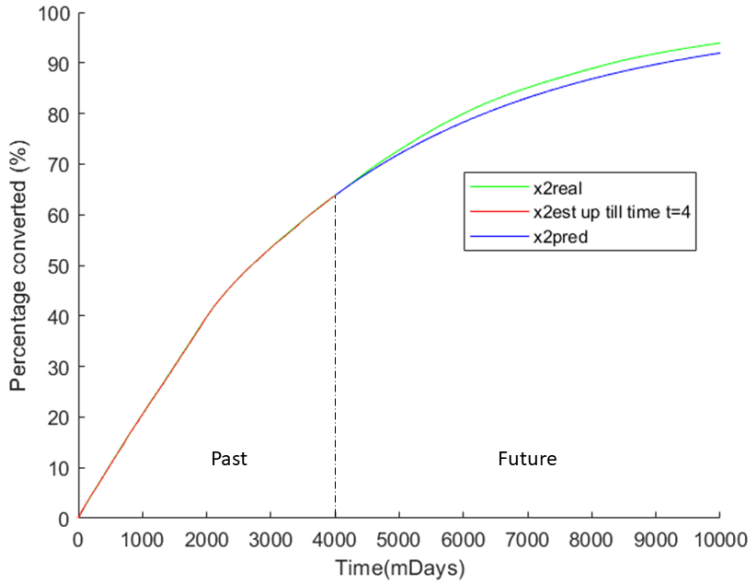

Figure 5-14: Prediction at time $\kappa = 4(\kappa = 4000$mDays) for mashing tank 1. In red the estimation up till time $\kappa = 4$, in yellow the real simulated data of the system, in blue the prediction upward of $\kappa = 4$.

result. As can be seen the prediction its trend is already at the beginning going down with respect to the prescribed process time of $p_1 = 7$, and converts after the time goes passed to the time-instance where the real data meats the threshold point. After the time-index reaches the process time, of course the predicted time will be equal to the estimated time of the threshold point, since the data up till the end-time of the process is received. This is only one realization, thus any hard conclusions can not be drawn. Therefore a montecarlo simulation with $n = 10$ is done for each of the individual subsystems for a trend be observed.

In the following tables each of the processes is ran 10 times, where the predictions from discrete time-instant $\kappa = 0$ are given up till $\kappa = 12$. With the process times and uncertainties somehow known, it is not most likely one of the process times reaches above the $\kappa = 12$. The prediction are rounded up to 1 decimal, also for the realization in the schedule. These roundings are made such that the process times not become to cluttered. In tables 5-7. 5-8, 5-9 and 5-10 the 10 realizations are shown for the 4 machines with deviating processing times. For $\kappa = 0$, the initial point, of course the processing times are all equal to the nominal processing times. As can be observed, almost all realizations convert with the predictions to the real

Realization	Ultimate p_1	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$
1	6.60	7.00	6.50	6.60	6.50	6.50	6.80	6.70	6.60	6.60	6.60	6.60	6.60	6.60
2	9.20	7.00	8.30	8.90	8.70	8.50	8.40	9.20	9.40	9.30	9.10	9.20	9.20	9.20
3	7.40	7.00	7.00	7.20	7.30	7.70	7.90	7.40	7.40	7.40	7.40	7.40	7.40	7.40
4	6.30	7.00	6.60	6.30	6.20	6.60	6.40	6.30	6.30	6.30	6.30	6.30	6.30	6.30
5	7.50	7.00	7.00	6.90	6.60	6.70	7.20	7.40	7.50	7.50	7.50	7.50	7.50	7.50
6	7.70	7.00	6.80	6.50	6.90	7.30	7.70	7.70	7.70	7.70	7.70	7.70	7.70	7.70
7	6.10	7.00	6.40	6.40	6.00	6.00	6.00	6.00	6.10	6.10	6.10	6.10	6.10	6.10
8	7.70	7.00	8.00	7.50	7.80	8.30	7.90	7.90	7.60	7.70	7.70	7.70	7.70	7.70
9	9.70	7.00	7.50	7.60	8.00	8.30	8.50	9.60	10.20	10.00	9.80	9.70	9.70	9.70
10	5.80	7.00	6.40	6.40	6.10	5.90	5.90	5.80	5.80	5.80	5.80	5.80	5.80	5.80

Table 5-7: 10 realizations of predictions made per discrete time-instance κ for the process in machine 1.

Realization	Ultimate p_2	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$
1	8.50	8.00	7.90	8.10	8.20	8.10	8.50	8.60	8.60	8.60	8.50	8.50	8.50	8.50
2	9.50	8.00	8.10	8.00	8.40	8.30	8.70	9.00	9.30	9.40	9.50	9.50	9.50	9.50
3	6.90	8.00	8.20	8.40	7.80	7.20	7.20	7.00	6.90	6.90	6.90	6.90	6.90	6.90
4	7.90	8.00	8.30	8.10	8.30	8.20	8.00	8.00	7.90	7.90	7.90	7.90	7.90	7.90
5	10.90	8.00	8.20	8.70	8.90	9.00	9.60	9.70	9.80	10.50	10.40	10.80	10.90	10.90
6	7.40	8.00	7.80	7.80	7.80	7.60	7.50	7.40	7.40	7.40	7.40	7.40	7.40	7.40
7	8.20	8.00	8.00	7.90	8.40	8.20	8.10	8.30	8.20	8.20	8.20	8.20	8.20	8.20
8	8.80	8.00	8.10	8.10	8.70	8.70	8.90	8.60	8.70	8.70	8.80	8.80	8.80	8.80
9	7.40	8.00	7.80	7.50	7.80	7.60	7.60	7.50	7.50	7.40	7.40	7.40	7.40	7.40
10	7.30	8.00	7.70	7.50	7.20	7.40	7.20	7.30	7.20	7.30	7.30	7.30	7.30	7.30

Table 5-8: 10 realizations of predictions made per discrete time-instance κ for the process in machine 2.

ultimate value of the processing time realization. Some realizations have some chattering in between but 2-3 time-units before the ultimate value of the process time, it already is very near to the ultimate process time. In other words, when coming closer to the real endtime of the process with respect to the discrete-time counter κ, the system becomes closer and closer to the real ultimate value. Furthermore, it is nice to observe the processing times where the ultimate value has a high difference from the nominal value, this can be recognized relatively quite fast. For instance, when looking at realization 5 in 5-8, the processing time becomes

Realization	Ultimate p_3	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$	$\kappa = 4$	$\kappa = 5$	$\kappa = 6$	$\kappa = 7$	$\kappa = 8$	$\kappa = 9$	$\kappa = 10$	$\kappa = 11$	$\kappa = 12$
1	5.30	6.00	6.80	6.10	5.60	5.50	5.30	5.30	5.30	5.30	5.30	5.30	5.30	5.30
2	7.50	6.00	6.50	7.40	8.00	7.30	7.90	7.80	7.50	7.50	7.50	7.50	7.50	7.50
3	5.60	6.00	5.80	5.80	5.70	5.70	5.60	5.60	5.60	5.60	5.60	5.60	5.60	5.60
4	6.10	6.00	6.20	6.30	6.50	6.30	6.20	6.10	6.10	6.10	6.10	6.10	6.10	6.10
5	5.30	6.00	6.00	5.70	5.40	5.30	5.40	5.30	5.30	5.30	5.30	5.30	5.30	5.30
6	5.20	6.00	5.80	5.90	5.40	5.20	5.20	5.20	5.20	5.20	5.20	5.20	5.20	5.20
7	8.60	6.00	6.70	7.50	8.90	8.60	8.40	8.40	8.50	8.70	8.60	8.60	8.60	8.60
8	5.70	6.00	5.60	5.70	5.60	5.50	5.60	5.70	5.70	5.70	5.70	5.70	5.70	5.70
9	4.90	6.00	5.50	5.10	4.80	4.90	4.90	4.90	4.90	4.90	4.90	4.90	4.90	4.90
10	6.30	6.00	6.00	5.80	6.20	6.30	6.30	6.30	6.30	6.30	6.30	6.30	6.30	6.30

Table 5-9: 10 realizations of predictions made per discrete time-instance κ for the process in machine 3.

Realization	Ultimate p_4	$\kappa = 0$	$\kappa = 1$	$\kappa = 2$	$\kappa = 3$	$\kappa = 4$	$\kappa = 5$	$\kappa = 6$	$\kappa = 7$	$\kappa = 8$	$\kappa = 9$	$\kappa = 10$	$\kappa = 11$	$\kappa = 12$
1	4.30	5.00	4.60	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30
2	3.80	5.00	4.40	3.80	3.90	3.80	3.80	3.80	3.80	3.80	3.80	3.80	3.80	3.80
3	6.50	5.00	5.60	6.30	6.10	6.40	6.40	6.60	6.50	6.50	6.50	6.50	6.50	6.50
4	5.40	5.00	5.10	5.10	4.90	5.20	5.40	5.40	5.40	5.40	5.40	5.40	5.40	5.40
5	6.50	5.00	5.70	6.00	6.50	5.90	6.40	6.50	6.50	6.50	6.50	6.50	6.50	6.50
6	4.30	5.00	5.00	4.60	4.40	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30
7	5.80	5.00	5.20	5.70	5.70	5.70	5.80	5.80	5.80	5.80	5.80	5.80	5.80	5.80
8	4.70	5.00	4.40	4.70	5.00	4.80	4.70	4.70	4.70	4.70	4.70	4.70	4.70	4.70
9	5.20	5.00	5.20	5.50	5.20	5.20	5.30	5.20	5.20	5.20	5.20	5.20	5.20	5.20
10	4.30	5.00	4.60	4.20	4.20	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30

Table 5-10: 10 realizations of predictions made per discrete time-instance κ for the process in machine 4.

10.90. It can already be observed at discrete time-instant $\kappa = 5$ this value becomes closer to 10. Of course this is useful for the scheduler, since it will most likely update when processing times have largely deviating values from the nominal processing times.

The above predictions are used in various ways for the upcoming schedulers. Since there will be 2 schedulers discussed with updating processing times, it will be shortly explained how the same kind of method is used in the updated schedulers. For the cyclic update behaviour per cycle k, the time-updates per discrete time-counter κ are not available. Therefore per cycle, the values in the column for "Ultimate p_i" for $i = 1, .., 4$ are used in the tables. For the time-updated scheduler, the values are used from $\kappa = 0$ up till the ceiled value of the ultimate processing times. In other words, when for example in machine 1 the processing time is $p_1 = 6.60$ the values are used up till $\kappa = 7$ since at $\kappa = 7$, all data became available such that the most accurate prediction is made, i.e. the estimation. Note that all the prediction are now done perfectly timed in an integer range where time-step κ is done per time-unit, but in the real system this is not necessarily the case. For instance, when taking route/mode 1 through machines 1 and 3. If machine 1 takes as an example $p_1 = 6.60$ to finish the process, the process in machine 3 starts at $\kappa = 6.60$. When arriving at the discrete time-counter $\kappa = 7$, the prediction became an estimation for the process in machine 1, but for the process in machine 3, a prediction can be made with the measurements up till $\kappa = 7$. Since the process started at $\kappa = 6.60$, over the time passed $\delta\kappa = 0.4$ measurements are obtained and a prediction can be made.

5-3 Scheduler

In the sections above, the simulations, estimations and predictions are made for the system. In this part of the report, the results for the scheduler are shown. The section is divided into 3 subsections. The statical scheduler, in which the initial optimization is discussed. The dynamical scheduler with respect to the cycle/job k, where the schedule is updated per cycle/job k. And at last the dynamical scheduler with respect to the discrete time-counter κ, in which the scheduler is optimized every time step.

5-3-1 Statical case

In the statical case, the scheduler is optimized with respect to all the initial values. First the scheduler is considered in which only routing takes place. Afterwards the scheduler is obtained in which routing and ordering take place to observe possible differences. The statical scheduler is also obtained for the comparison with the cases discussed below, though then with the implemented adjusted times of the subsystems. The input $u(k)$ for all the schedulers in this thesis, also the dynamical ones, is taken as 0 for all k. In the cost function therefore from 4-12, it can be left out. The first reason to leave out the input, is to make sure differences in the schedule are not related to a too late incoming input batch, when the input is given beforehand. The second reason, is that if the input is made as a variable as well, the algorithm becomes computationally too complicated, where as a results, the optimizations take too long. The input is then considered as a non-binary variable namely. As a third reason, in a realistic way, input time-instances are not very likely to be chosen or tuned precisely, more to be filled in according to deliveries of other parties which in general can not be chosen exactly beforehand. In other words, one chooses a delivery of specific resource, but the delivery can only be chosen to a specific time-span, such that it is not freely tunable.

In the routing case, the cycles need to be scheduled after one another for all jobs. This means the jobs in machine 5, since they all need to undergo machine 5, will be processed in consecutive order in which they started at the beginning of the process. Note that the modelling of this kind of schedulers is not preferable, since if job 1 has a very significant delay, all jobs need to wait for job 1 to come to machine 5. The schedule is shown in figure 5-15. As can be observed nicely, all the jobs start as soon as the previous process in the specific job is finised. In other words, all the individual jobs, each coloured block, starts in the distinct machine precisely after the same coloured block in another machine is done. It can also be observed that the machines 1 and 2 are constantly working. The modes/routes and endtimes of each of the jobs is summarized in table 5-11. The sum of the end-times, i.e. the value of the cost function, is equal to FVAL = 202.

Cycle/job	1	2	3	4	5	6	7	8
Modes/routes	2	3	1	4	1	4	1	4
Endtime	13.00	15.00	21.00	22.00	28.00	30.00	35.00	38.00

Table 5-11: Table of the routes chosen per job and the endtime per job for the statical scheduler when only routing is allowed.

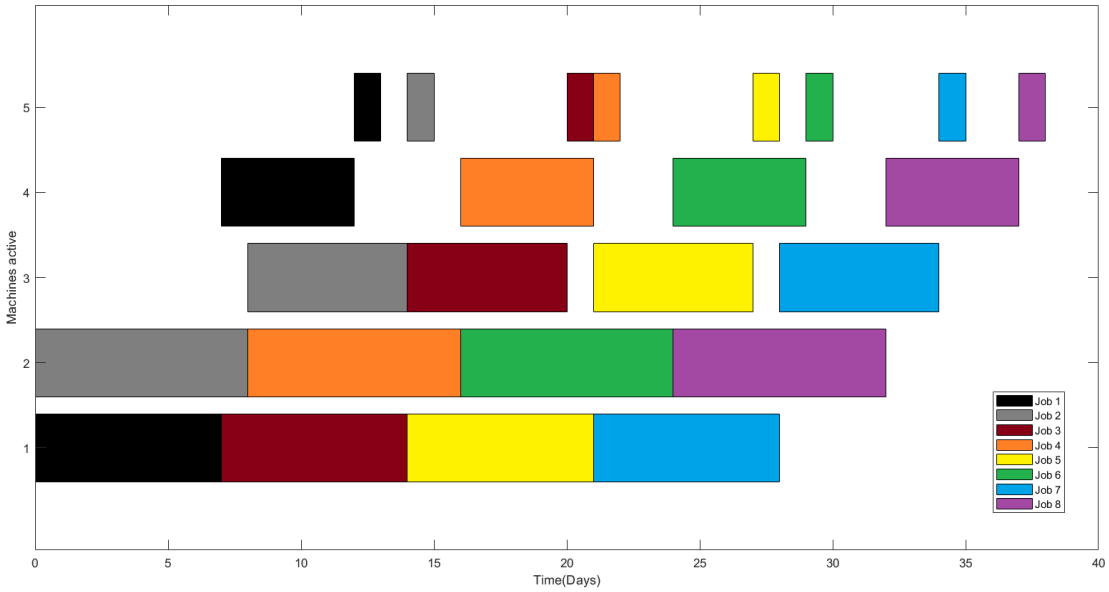

Figure 5-15: GANTT chart for the scheduler when only routing of the jobs is considered. On the y-axis the bottom level machine M_1 to the top level machine M_5.

For the next figure, the statical scheduler is shown with ordering involved. Essentially ordering in a statical way will not influence the outcome that much, since the routing already took care of the minimal outcome of the jobs after one another and therefore still needs to assign the jobs to a specific cycle. In other words, it could assign job 3 to go route 3, and job 4 to go route 4, but also job 3 to go route 4 and job 4 to go route 3, such that ordering already occurs a little bit in the routing scheduler. The only thing the ordering process could add, is that job 3 is chosen to have route 3, job 4 chosen to have route 4, but the process in machine 5 in job 4 goes before the process in machine 5 of jobs 3. The results for the GANTT-chart of the statical scheduler with the ordering in machine 5 involved is shown in figure 5-16. The modes and endtimes for the case of the statical scheduler with ordering involved are shown in table 5-12. It can be observed ordering takes place in a lot of jobs since the endtimes of

Cycle/job	1	2	3	4	5	6	7	8
Modes/routes	3	3	2	3	2	2	3	2
Endtime	15.00	23.00	13.00	31.00	20.00	27.00	39.00	34.00

Table 5-12: Table of the routes chosen per job and the endtime per job for the statical scheduler when routing and ordering is allowed.

jobs in future cycles are before the endtimes of jobs in previous cycles, this could never be the case in the routing example. The sum of the endtimes however is equal to $\sum_{end} = 202$ and exactly the same as the routing case. This is reasonable, since the routing process can order its processes at the beginning of the optimizations since the routes are not yet assigned to each of the jobs. Though, note that this also means not a unique solution exists to the most optimal initial schedule.

The optimality or benefits from the statical scheduler can be discussed by assigning ran-

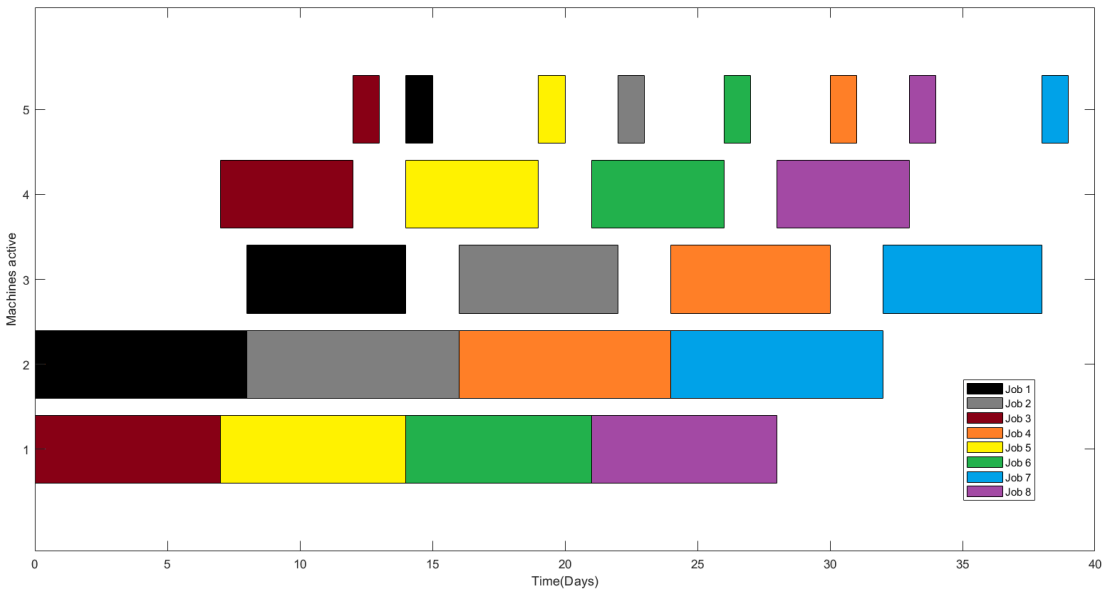

Figure 5-16: GANTT chart for the scheduler routing and ordering of the jobs is considered. On the y-axis the bottom level machine M_1 to the top level machine 5 M_5.

dom modes to the statical scheduler, and let it simulate itself with the constraints discussed in the methods section. The results of a montecarlo-simulation with $n = 100$ for the value of the cost-function are shown in figure 5-17. As can be observed the results are much higher

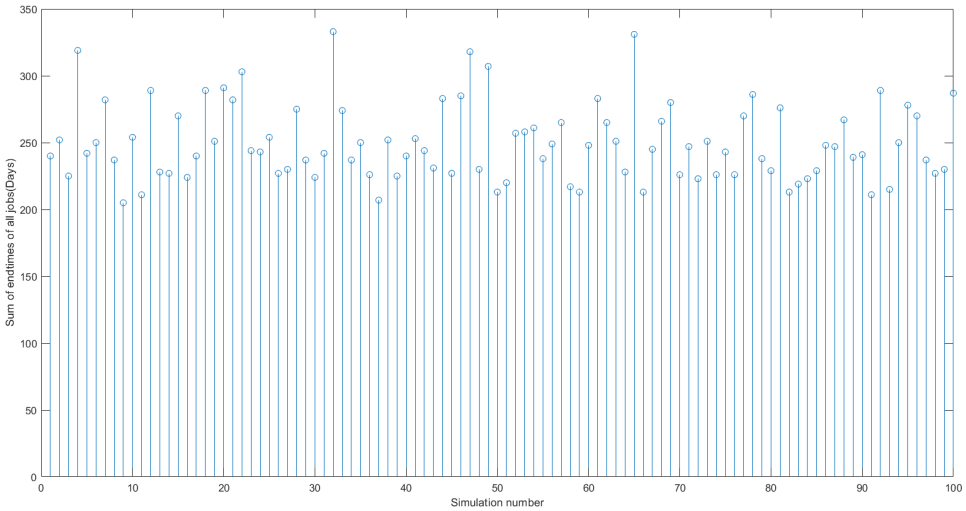

Figure 5-17: Monte-carlosimulation for n=100 of random modes assigned to the optimization algorithm.

than the sum of the optimized schedule, no matter the routing or ordering case, where the sum of the endtimes was equal to $\sum_{end} = 202$. To mathematically substantiate this conclusion, the mean and standard deviation of the randomly assigned route variables are calculated of

figure 5-17. These are equal to $\mu(\sum_{end}^{total}) = 249.71$ and $\sigma(\sum_{end}^{total}) = 28.5$. Of course we can repeat the process for a larger number of simulations, but as we will see, this gives the same results of a large difference from the optimal value of the schedule $\sum_{end} = 202$. For the next paragraphs, the statical scheduler is taken as a comparison for the dynamical cases to observe the improvements and benefits.

5-3-2 Dynamical case - Event-based update k

The dynamical case with respect to the cycle k will update the schedule by receiving new information every cycle it increases. The cases discussed are of disruption, certain cycle-instances where the process time of one of the processes increases dramatically, and disturbances, the process times are updated by the adjusted process time due to the uncertainties of noise as is discussed in the chapter about the subsystems. For better conclusions to make, the disturbance cases are subdivided into 3 approaches. In the first approach, only machine 1 and 2 are subjected to deviating processing times. In the second approach, only machine 3 and 4 are subjected to deviating processing times. At last, all machines except machine 5 are subjected to deviating processing times.

Disruptions

In this case, a certain mode is chosen, whereafter it happens that in this specific route, one of the machines encounters problems where the process time increases dramatically. The point of this case, is to show how the scheduler can adapt to certain drastical changes in the process time. For the first scenario, machine 4 is chosen to have process times $p_4(k) = 35$ for $k = 1, 2$. The scheduler receives this new information, when the route is already chosen for this mode. In other words, if $p_4(k) = 35$ for $k = 1$ the scheduler receives this information at cycle-instant $k = 2$. In figure 5-18 the GANTT-chart is shown for the optimal schedule where the optimization is done at cycle-instant $k = 1$. Now if we go one step further in the process, at $k = 2$, the scheduler gets information about the process time in machine 4 being $p_4(1) = 35$ for cycle $k = 1$. Of course, the scheduler changes the routes of the upcoming jobs to not go through machine 4 anymore, since the starting time has to be larger than the endtime of the job 1 in machine 4, which has a process time of 35. For that reason the process time of $p_4(2) = 35$ is not encountered in the model since the second job, job 2, will not be routed on a route through machine 4. As can be observed brilliantly in figure 5-19 the schedule adapts to the sake of the process time of machine 54 being $p_4(1) = 35$. If you look at job 1, clearly it has a much larger stroke in the machine 4 in figure 5-19. The result is that the total jobs taking place in machine 4 decreased from 5 to 2, and the total jobs take place in machine 3 increased from 3 to 6. Furthermore if we look at the starting time of job 1 in machine 5, we see in figure 5-18 there are no jobs before job 1, while in figure 5-19 we can observe 5 jobs in front of job 1 in machine 5, where it can be concluded the adaptive behaviour of the scheduler does function, also with respect to the ordering. To compare the results in terms of the cost function, the sum of the end-times is calculated, for the adjusted case with the updated process times, and the case where the schedule is following the initial routing and

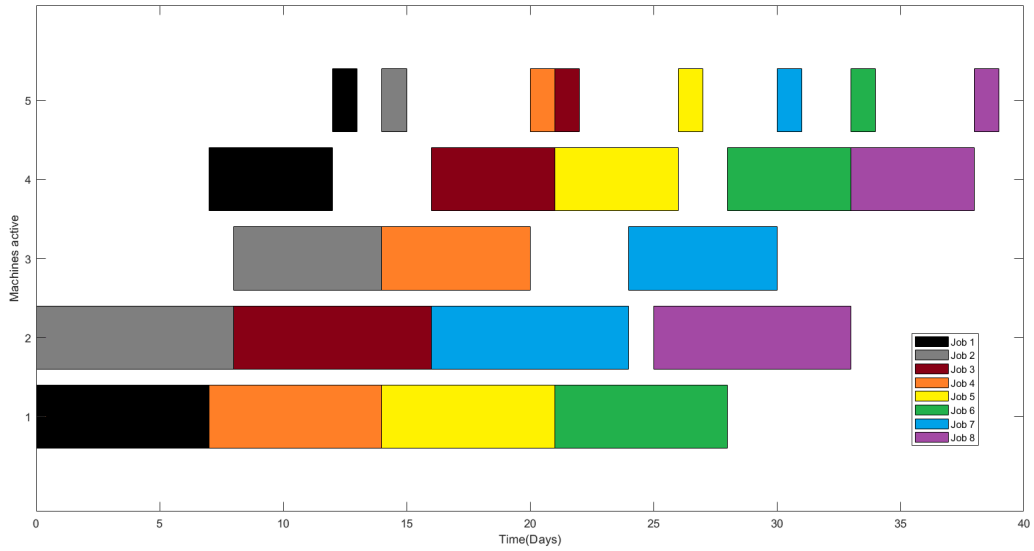

Figure 5-18: GANTT-chart of the initial schedule, where the scheduler does not yet know $p_4(1) = 35$. The initial schedule is based on the optimization when the scheduler is in cycle $k = 1$.

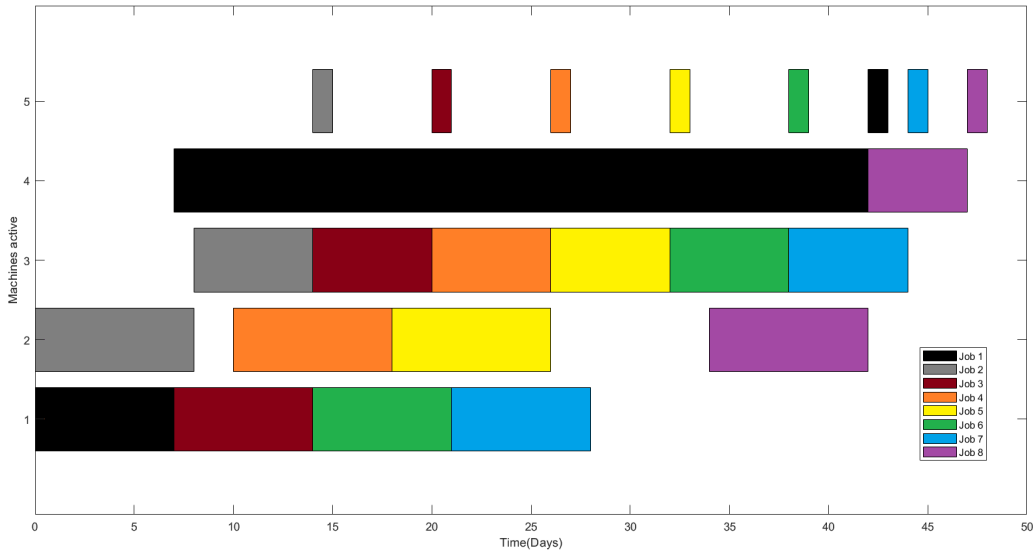

Figure 5-19: GANTT-chart of the updated schedule at cycle-instant $k = 2$ with updated information of process time $p_4(1) = 35$

ordering with the updated process times:

$$\begin{aligned} &\text{Situation 1 for initial schedule with updated process times } \textstyle\sum_{j=0}^{N_p} y(k+j) = 372 \\ &\text{Situation 2 for updated schedule with updated process times } \textstyle\sum_{j=0}^{N_p} y(k+j) = 230 \end{aligned} \tag{5-5}$$

In the results above it is nicely observed how the schedule can adapt with the benefit of the model becoming less late when minimizing over the cost function, i.e. all jobs need to be

finished as soon as possible.

To conclude this paragraph, one more example is shown to observe one of the disadvantages of the approach of updating the scheduler with respect to cycle k. In figure 5-20, the initial schedule is shown again. Now, we apply a drastical change in process time of $p_1(k) = 35$ for cycle $k = 6$. Of course the schedule wants to update and avoid loss of time due to the drastically changed process time of machine 1. However, the information about this drasti-

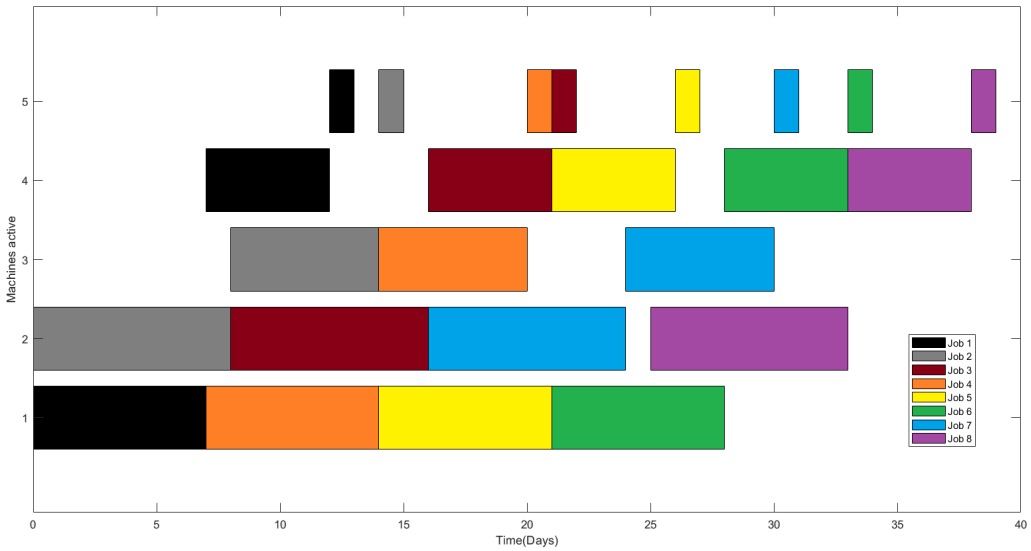

Figure 5-20: GANTT chart of the initial schedule where it is not yet known $p_4(10) = 35$.

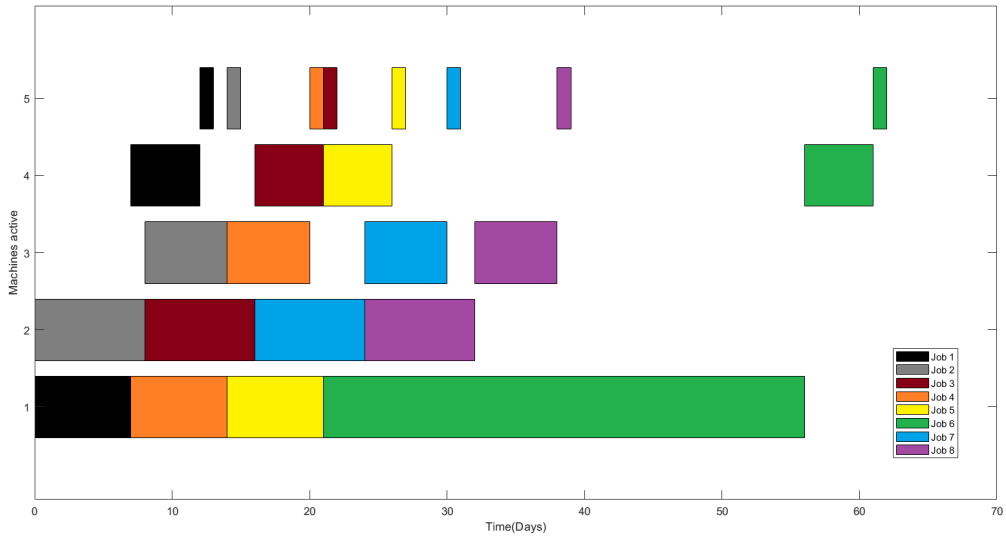

Figure 5-21: GANTT chart of the addaptive schedule where $p_4(10) = 35$ is known.

cally changed process time, comes in after the nominal endtime of the original process. When looking at job 6 the green block in machine 1 in figure 5-20, the nominal endtime(the end of the green block in machine 1) is at $\kappa = 28$. At $\kappa = 28$ we know that the process takes a

lot longer, such that we can adapt the schedule. If we now look at job 7 the purple block in 5-20, it starts at $\kappa = 25$ in machine 2, thus it is already started before job 6 the green block in machine 1 is nominally finished. If we now look at the updated schedule in 5-21, we can observe the starting time of job 7 the purple block in machine 2, has been shifted to the left where it starts at $\kappa = 24$. This is however never realistically possible, since it has already been started at $\kappa = 25$ before the nominal endtime of job 6 in machine 1 at $\kappa = 33$. The schedule has updated itself in the past. For that reason, cyclic updated behaviour is often not realistic, due to processes within the cycle have been already started with respect to time, and are therefore fixed. This conclusion opens way for the time-updated schedule, in which processes in each cycle are fixed with respect to time, instead of all processes are fixed with respect to the cycle they are in.

Disturbances - Machine M_1 and M_2

This is the first approach when having disturbances into the system. In the next cases, In figures 5-22 and 5-23 the results are shown. In figure 5-22 the results are shown for the initial scheduler with fixed order and routing but with updated processing times. This looks rather weird, for instance, when looking at the empty space in machine 5 for job 6, the green block, when noting that this job has already finished in machine 4. This is the reason why the updated scheduler can have a beneficial effect. It can be observed clearly, job 7, the job in

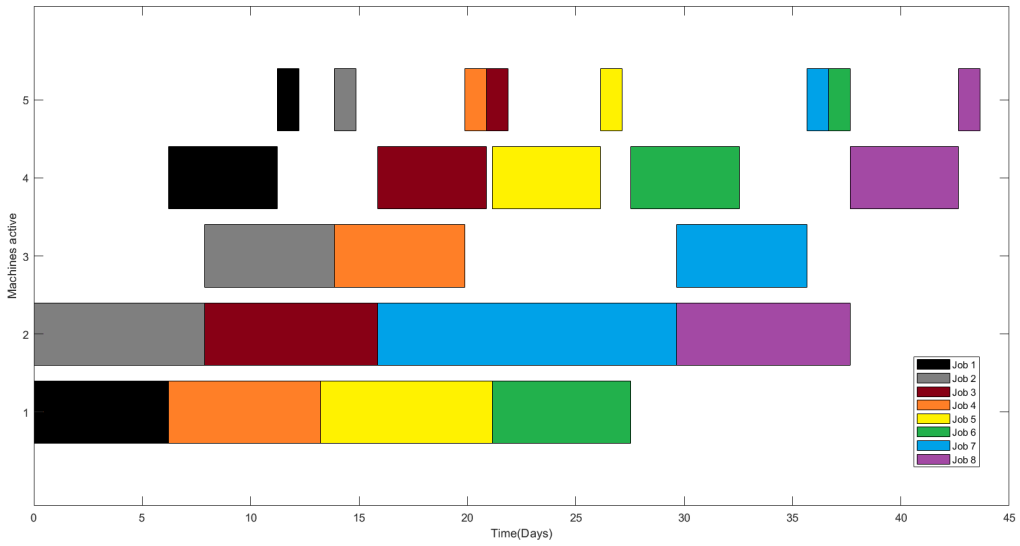

Figure 5-22: GANTT chart of the initial schedule with fixed routing and ordering but updated process times.

light blue, is having a tough time in machine 2. The bar in machine 2 is much longer than the other bars. However, in figure 5-22 it can be observed that the green block, job 6, is initially ordered after the light blue block, job 7, in machine 5. Therefore it has to wait for job 7 to be finished in machine 5. Furthermore, job 8, the purple block, is scheduled after job 7 in machine 2 as can be also observed in 5-22, while it could already start in machine 1. The ordering between job 6 and 7 in machine 5, as well as the routing of job 8 to machine

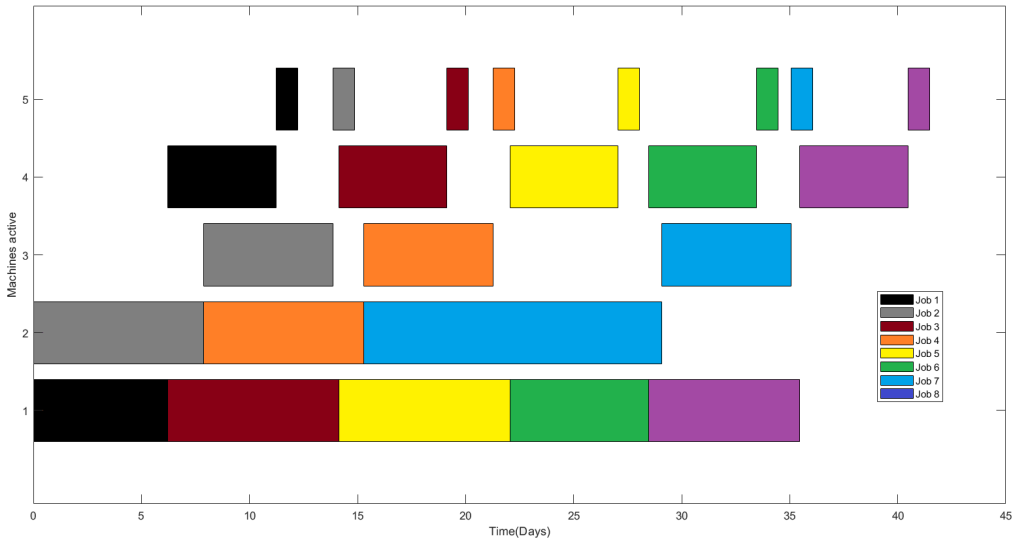

Figure 5-23: Final schedule for the first approach with respect to the cyclic updated scheduler in which only machine 1 and 2 have deviating proccessing times.

1 instead of machine 2, can be observed in the updated schedule. The updates of the modes can be observed in table 5-13. Furthermore, the realization of the ultimate process times per cycle/job k are given in table 5-14. In the beginning of this chapter, it is briefly discussed

	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$	$\ell(6)$	$\ell(7)$	$\ell(8)$
Routes at initial schedule $k = 1$	2	3	4	1	2	2	3	4
Routes at updated schedule $k = 8$	2	3	2	3	2	2	3	2

Table 5-13: Initial routes chosen by the scheduler versus the final routes chosen by the scheduler.

$p_1(k)$	$p_2(k)$
6.20	8.00
7.00	7.90
7.90	8.00
7.00	7.40
7.90	8.00
6.40	8.00
7.00	13.80
7.00	8.00

Table 5-14: Updated process times for the processes of machine 1 and 2.

how the scheduler should be compared. Now, the comparison is a lot more elaborated by the sake of some tables. In table 5-15 the differences between the updated process times and nominal process times are given. The updated processing times are only given for the routes the updated scheduler has been through, because it only starts a realization when the job/cycle is started in a specific machine and the scheduler is already passed it in its cyclic update. The numbers in bold given in table 5-15 are the ones the machines go through with

	$p_1(k) - p_1$	$p_2(k) - p_2$
	-0.80	0
	0	**-0.10**
	0.90	0
	0	**-0.60**
	0.90	0
	-0.60	0
	0	**5.80**
	0	0
$\sum$	+0.40	+5.10

Table 5-15: Differences between the updated ultimate process times with respect to the nominal process times. The routes taken by the final scheduler in bold.

the routing of the final schedule. Note that machines 3 and 4 are not obtained in this table since they do not deviate in process times in the first approach. Also, the last machine does not have an update, since at $k = 8$ the schedule is already fixed and it would not have a better outcome to the adaptive behaviour of the scheduler when also updating with respect to $k = 9$ since the schedule is already fixed at $k = 8$. Now we want to compare the results above with the routes taken by the initial schedule such that we can quantify what gains/losses the final scheduled got with the updated schedule with respect to the process times. In table 5-16 the numbers in bold are the routes taken by the initial scheduler. Note that the routes which are taken by the initial schedule but not by the updated schedule, are given with the nominal process times. Comparing the gains losses of the final schedule with the initial schedule now

	$p_1(k) - p_1$	$p_2(k) - p_2$
	-0.80	0
	0	**-0.10**
	0.90	**0**
	0	-0.60
	0.90	0
	-0.60	0
	0	**5.80**
	0	0
$\sum$	-0.50	+5.70

Table 5-16: Differences between the updated ultimate process times with respect to the nominal process times. The routes taken by the initial scheduler in bold.

is just simply adding up the sums of the deviations of processing times with respect to the nominal processing times. For the initial schedule this is $\sum_{total} = -0.50 + 5.70 = +5.20$ and for the final schedule this is $\sum_{total} = +0.40 + 5.10 = +5.50$. It can now be concluded that only due to the process times, the final scheduler will take $\sum_{dif} = +5.50 - 5.20 = +0.30$ longer. If we now look at the value of the cost function, the sum of the endtimes of all jobs, the final schedule even with longer sums of process times, does descrease the sum of endtimes of all jobs, as can be shown in table 5-17.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule	206.90	+0.30	207.20
Updated schedule	201.50	+0.30	201.50

Table 5-17: Value of the cost function, the sum of endtimes of all jobs, for the initial schedule and the final schedule.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule(1)	206.40	+2.90	209.30
Updated schedule(1)	203.80	+2.90	203.80
Initial schedule(2)	189.60	-1.50	188.10
Updated schedule(2)	187.80	-1.50	187.80
Initial schedule(3)	199.70	+0.80	200.50
Updated schedule(3)	197.00	+0.80	197.00
Initial schedule(4)	195.30	+1.70	197.00
Updated schedule(4)	194.30	+1.70	194.30
Initial schedule(5)	188.60	-0.40	188.20
Updated schedule(5)	188.30	-0.40	188.30
Initial schedule(6)	192.30	-3.00	189.30
Updated schedule(6)	184.10	-3.09	184.10

Table 5-18: 6 realizations of the updated scheduler with machine 1 and 2 deviating and the cost function for the initial schedule versus the final schedule shown.

By repeating the steps for 6 realizations, we can observe what is the trend or effect from the updated scheduler. The results are shown in table 5-18. What we carefully observe, is that the updated schedule has a beneficial effect for the cyclic updated scheduler when only machines 1 and 2 are obtained to deviating processing times, but just for only a few time-units. However, in realization 5, it can be observed the initial schedule has a better outcome than the final schedule. The most suitable explanation, is that it could occur when a system switches to another route for a specific job, afterwards the realization is made and turned out to be better for the non-updated schedule. The updated schedule can not switch back anymore, and therefore has a higher cost.

Disturbances - Machine M_3 and M_4

To continue the cyclic updated behaviour, the second approach is discussed, where only machines 3 and 4 are subjected to deviating processing times. The approach for the analysis of this approach is quite the same as for the first approach where machines 1 and 2 have deviating processing times.

In figure 5-24 and 5-25 the results are shown for the initial scheduler with updated process time, and the updated schedule with updates per cycle k. It can be observed, the scheduler updated with respect to its routes for each of the jobs. It seems the scheduler rescheduled when observing job 2, the gray block is taking a little bit longer than expected in machine 3. This is caused due to the fact in figure 5-24 the orange block, job 4, after the gray block, job 2, in machine 3 will cause the orange block in machine 5 to be delayed. If the orange block

in machine 5 is delayed, it can be observed in figure 5-24 the red block, job 3, need to wait in machine 5 while it is already finished in machine 4, this means timeloss. It is solved by replacing job 4, the orange block in machine 3 by job 3 the red block as can be observed in 5-25. To do this, job 5, the yellow block, is rescheduled in front of job 3, the red block, in machine 4 as can be concluded when comparing 5-24 and 5-25. Note that the yellow block, job 5, can be seen as shifting itself in front of the red and orange block, respectively job 3 and 4, when comparing the figures. However, one could ask themselves, why not make the yellow block in figure 5-25 the orange block, since the update is made at $k = 2$ as can be observed since job 3, the red block is changed from its route. This is only possible when the job is not yet passed, i.e. the mode changed at $k = 2$. This is simply explained that for the optimization it does not matter at a specific cycle-instant k since the value of the cost function will be the same. Though, it is preferable to have the jobs in a chronological order, since they are updated in consecutive order. This is something to look at in further research. To compare

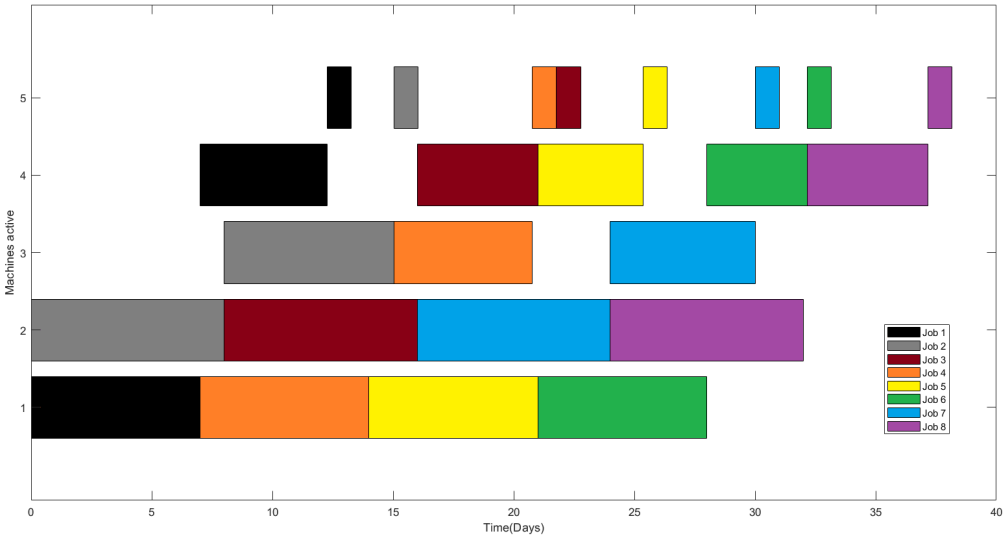

Figure 5-24: GANTT chart of the initial schedule with fixed routing and ordering but updated process times.

	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$	$\ell(6)$	$\ell(7)$	$\ell(8)$
Routes at initial schedule $k = 1$	2	3	4	1	2	2	3	4
Routes at updated schedule $k = 8$	2	3	3	3	2	2	2	4

Table 5-19: Initial routes chosen by the scheduler versus the final routes chosen by the scheduler.

the results, the same procedure is done as in the approach previously described for deviating process times with respect to machine 1 and 2. The mode changes are given in table 5-19. In tables 5-20 and 5-21 the values for the updated process times and the differences of the updated process times with the nominal process times are given. To compare one another, we make the same approach as previously done for the previous approach. If we repeat the same procedure we can come up with $\sum_{total} = -0.50 - 1.80 = -2.30$ for the final scheduler compared with $\sum_{total} = +0.80 - 1.10 = -0.30$ for the initial schedule such that only due to the process times $\sum_{dif} = -2.30 - (-0.30) = -2.00$ the final schedule is already 2 time-units

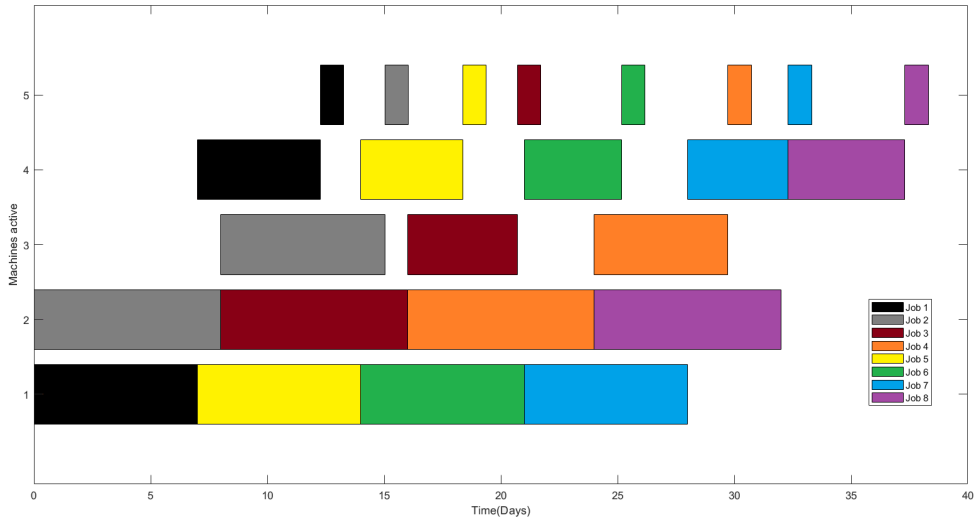

Figure 5-25: Final schedule for the second approach with respect to the cyclic updated scheduler in which only machine 3 and 4 have deviating processing times.

$p_3(k)$	$p_4(k)$
6.00	5.30
7.10	5.00
4.70	5.00
5.70	5.00
6.00	4.40
6.00	4.20
6.00	4.30
6.00	5.00

Table 5-20: Updated process times for the processes of machine 3 and 4.

	$p_3(k) - p_3$	$p_4(k) - p_4$
	0	**0.30**
	1.10	0
	-1.30	0
	-0.30	0
	0	**-0.60**
	0	**-0.80**
	0	**-0.70**
	0	**0**
$\sum$	-0.50	-1.80

Table 5-21: Differences between the updated ultimate process times with respect to the nominal process times. The routes taken by the final scheduler in bold.

faster. Summarizing the results in table 5-23. We can now retrieve that the updated schedule with respect to the initial schedule gaines 2 time-units when looking at the sum of the

	$p_3(k) - p_3$	$p_4(k) - p_4$
	0	**0.30**
	1.10	0
	-1.30	**0**
	-0.30	0
	0	**-0.60**
	0	**-0.80**
	0	-0.70
	0	**0**
$\sum$	+0.80	-1.10

Table 5-22: Differences between the updated ultimate process times with respect to the nominal process times. The routes taken by the initial schedule in bold.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule	194.50	-2.00	192.50
Updated schedule	190.80	-2.00	190.80

Table 5-23: Value of the cost function, the sum of endtimes of all jobs, for the initial schedule and the final schedule.

end-times of all jobs when only deviating the process times of machine 3 and 4. If we repeat this procedure for 6 realizations we get the values presented in table 5-24. When looking at

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule(1)	197.00	+2.10	199.10
Updated schedule(1)	195.40	+2.10	195.40
Initial schedule(2)	197.10	+0.80	197.90
Updated schedule(2)	195.60	+0.80	195.60
Initial schedule(3)	211.70	+1.60	213.30
Updated schedule(3)	202.10	+1.60	202.10
Initial schedule(4)	194.50	-1.10	194.50
Updated schedule(4)	192.70	-1.10	192.70
Initial schedule(5)	197.10	+1.00	198.10
Updated schedule(5)	195.50	+1.00	195.50
Initial schedule(6)	194.30	-0.70	193.60
Updated schedule(6)	193.20	-0.70	193.20

Table 5-24: 6 realizations of the updated scheduler with machine 3 and 4 deviating and the cost function for the initial schedule versus the final schedule shown.

the results, it can be observed that some of the cases are well improved with the updated schedules. For the sharp eye, it looks a little bit like the major improvements, can be found in the realizations where FVAL is high. This can be explained by the fact, if FVAL is high, a lot of processing times are delayed. By looking at $\sum_{dif}$ being not so great, it can be concluded the initial schedule takes the same machines as the final schedule where the process times are much delayed. Ultimately, the adaptive scheduler adapts to this large process times by reordering in machine 5, while the initial schedule need to follow its fixed order. This is a

possible explanation for the improvements of FVAL when having a high FVAL to begin with, and not a too high or low $\sum_{dif}$.

Disturbances - All machines

For the last approach, all machines are having deviating process times except for machine 5. Machine 1 and 2 for the mashing processes, machine 3 and 4, for the brewing processes. One realization is illustrated whereafter again 6 realizations are compared with the cost function. In figures 5-26 and 5-27 the GANTT-charts are given for the initial schedule with updated process time but fixed routing and order, and the final schedule with updated process time. Note that the horizontal axis in both figures has a different scale, where it looks like the end-times of the processes from the final schedule arrive later in machine 5 than the processes in the initial schedule where this is not the case. Also the blocks might seem longer than one another, while they have the same duration but in a different scale. As can be observed

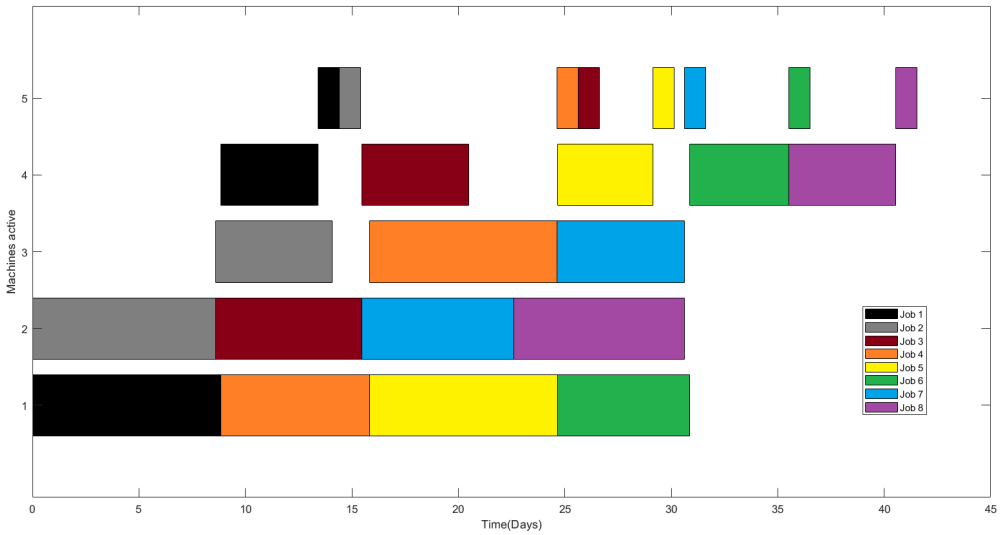

Figure 5-26: GANTT chart of the initial schedule with fixed routing and ordering but updated process times.

brilliantly the red block in figure 5-26 is losing a lot of time when looking at the end times in machine 4 and the starting time in machine 5. Since it is the initial ordering, it needs to stay behind the orange block, block 4. The updated behaviour of the routes is summarized in table 5-25 and also observed in figure 5-27. As can be seen the red block is ordered in front of the orange block. Note that the update for the red block is before the update of the orange block, the red block namely is job $k = 3$ where the orange block is job $k = 4$, such that at the update of job/cycle $k = 3$, the system did not yet know the orange block was going to take such a long time. To conclude, the red block is not shifted in front of the orange block, due to the process times of the orange block. The red block is placed in front however due to the processing times of the gray and black block in machines 1 and 2, respectively job 1 and 2. Furthermore, when the orange block did experience such long processing times in

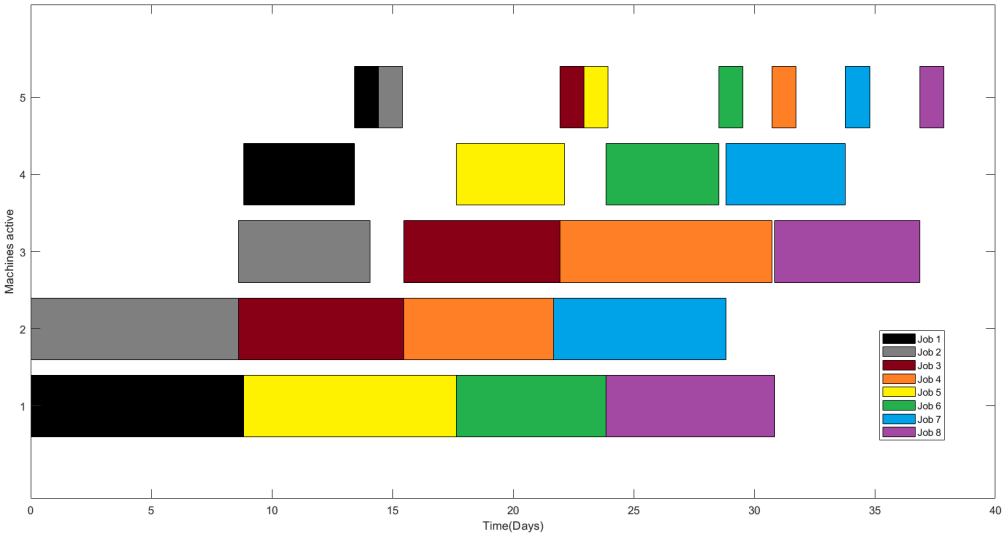

Figure 5-27: Final schedule for the third approach with respect to the cyclic updated scheduler in which all machines have deviating processing times.

machine 3, the yellow and green block, respectively job 5 and 6, are placed in front of the orange block due to the adapted behaviour of the scheduler. To continue the same procedure,

	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$	$\ell(6)$	$\ell(7)$	$\ell(8)$
Routes at initial schedule $k=1$	2	3	4	1	2	2	3	4
Routes at updated schedule $k=8$	2	3	3	3	2	2	4	1

Table 5-25: Initial routes chosen by the scheduler versus the final routes chosen by the scheduler.

we will now look at the updated processing times as have been shown in table 5-26. And

$p_1(k)$	$p_2(k)$	$p_3(k)$	$p_4(k)$
8.80	8.00	6.00	4.60
7.00	8.60	5.50	5.00
7.00	6.90	6.50	5.00
7.00	6.20	8.80	5.00
8.80	8.00	6.00	4.50
6.20	8.00	6.00	4.70
7.00	7.10	6.00	5.00
7.00	8.00	6.00	5.00

Table 5-26: Updated process times for the processes of machine 1, 2, 3 and 4.

applying the same method for the sum of the deviation with respect to the nominal process times. Such that for the final scheduler $\sum_{total} = +2.80 - 3.20 + 2.80 - 1.20 = +1.20$ and for the initial scheduler $\sum_{total} = +2.80 - 1.40 + 2.30 - 1.20 = +2.50$ such that the difference is $\sum_{dif} = +1.20 - 2.50 = -1.30$. The final scheduler by means of the updated process times in general is therefore 1.30 time-units faster. Repeating the same procedure, the result is shown in table 5-29. Repeating this step for 6 realizations gives the following table 5-30. As can

	$p_1(k) - p_1$	$p_2(k) - p_2$	$p_3(k) - p3$	$p_4(k) - p4$
	1.80	0	0	**-0.40**
	0	**0.60**	**-0.50**	0
	0	**-1.10**	**0.50**	0
	0	**-1.80**	**2.80**	0
	1.80	0	0	**-0.50**
	-0.80	0	0	**-0.30**
	0	**-0.90**	0	**0**
	0	0	**0**	0
$\sum$	+2.80	-3.20	+2.80	-1.20

Table 5-27: Differences between the updated ultimate process times with respect to the nominalprocess times. The routes taken by the final scheduler in bold.

	$p_1(k) - p_1$	$p_2(k) - p_2$	$p_3(k) - p3$	$p_4(k) - p4$
	1.80	0	0	**-0.40**
	0	**0.60**	**-0.50**	0
	0	**-1.10**	0.50	**0**
	0	-1.80	**2.80**	0
	1.80	0	0	**-0.50**
	-0.80	0	0	**-0.30**
	0	**-0.90**	**0**	0
	0	**0**	0	**0**
$\sum$	+2.80	-1.40	+2.30	-1.20

Table 5-28: Differences between the updated ultimate process times with respect to the nominalprocess times. The routes taken by the initial schedule in bold

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule	213.90	-1.30	212.60
Updated schedule	202.60	-1.30	202.60

Table 5-29: Value of the cost function, the sum of endtimes of all jobs, for the initial scheduleand the final schedule.

be observed. Almost all schedulers have a beneficial effect on the total endtimes of the jobs when adapting. Some remarkable things can be observed. To begin with the cases where $\sum_{dif} = +0.00$. This could mean the schedule is exactly the same as the initial schedule but has a different order in machine 5 due to the changing processing times. It could also be a coincidence when the machine have taking other processes who eventually deviate, but where the sum of the deviations equals 0. The case where $\sum_{dif} = +7.90$, it can be nicely observed that even when the routes chosen have a total difference of +7.90 in total processing times compared with the initial schedule, the final schedule still remains close to the initial schedule when comparing the FVAL 231.80 and 233.60. The overal outcome can be observed as that the scheduler which updates when all machines have deviating processing times, will have a beneficial effect on the sum of endtimes of all jobs. A montecarlo simulation can be done to substantiate this for a very large number of simulations. The results are shown in table

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule(1)	218.40	-2.00	216.40
Updated schedule(1)	206.40	-2.00	206.40
Initial schedule(2)	201.90	+2.40	204.30
Updated schedule(2)	204.80	+2.40	204.80
Initial schedule(3)	188.90	+0.00	188.90
Updated schedule(3)	185.50	+0.00	185.50
Initial schedule(4)	206.10	-1.10	205.00
Updated schedule(4)	200.60	-1.10	200.60
Initial schedule(5)	231.80	+7.90	239.70
Updated schedule(5)	233.60	+7.90	233.60
Initial schedule(6)	220.90	+1.30	222.20
Updated schedule(6)	212.00	+1.30	212.00

Table 5-30: 6 realizations of the updated scheduler with all machines deviating and the cost function for the initial schedule versus the final schedule shown.

5-31. In the table, the averages of respecectively the value of the cost for the final schedule,

$\text{FVAL}_{new} - FinalScheduler$	$\text{FVAL}_{new} - Initialscheduler$	$\sum_{dif}$	Decrease in cost(%)
199.97	203.00	+0.71	-1.21%

Table 5-31: Montecarlo simulation of the cycle-based updated scheduler with $n = 100$ with average value of the final scheduler its cost, average value of the initial scheduler its cost, average $\sum_{dif}$ and average decrease in total endtime with respect to the initial schedule in %.

the value of the cost for the initial schedule, the $\sum_{dif}$ and the decrease in % with respect to the initial schedule its cost. To conclude, from the montecarlo simulation a decrease can be observed. However, the decrease is marginal.

5-3-3 Dynamical case - Discrete time-based update κ

As was concluded in the previous paragraph. The cyclic updated scheduler does not always work realistically. This is caused by the cylic updated behaviour where per cycle the scheduler is updated, while time could already be passed future cycle processes. To solve this, a time-based updated scheduler is obtained. In the following paragraphs first the time based scheduler is subjected to disruptions. Afterwards, the timed-based scheduler is subjected to the disturbances of the processes where the same cases are obtained as in the cyclic-updated scheduler analysis. First machine 1 and 2 are subjected to disturbances, afterwards machine 3 and 4, and ultimately all machines with exception of machine 5 are subjected to disturbances.

Disruptions

The main difference for the disrupted case in the time-updated scheduler, is that the mode is not fixed when time increases. Only processes are fixed when the time passed the starting time. This means for instance when mode $\ell(k) = 1$ is chosen, and the time passed the starting time for process 1, but not for process 3, the system can still change its route from $\ell(k) = 1$ to $\ell(k) = 2$. This can be observed brilliantly with the next case, where a disruption takes place at machine 1 and 2 as $p_1(k) = 35$ and $p_2(k) = 35$ for $k = 1$. In figures 5-28 and 5-29, the 2 cases where the initial schedule with initial order and routes is used with the updated processing times, and the second case where the updated order and routes are obtained.

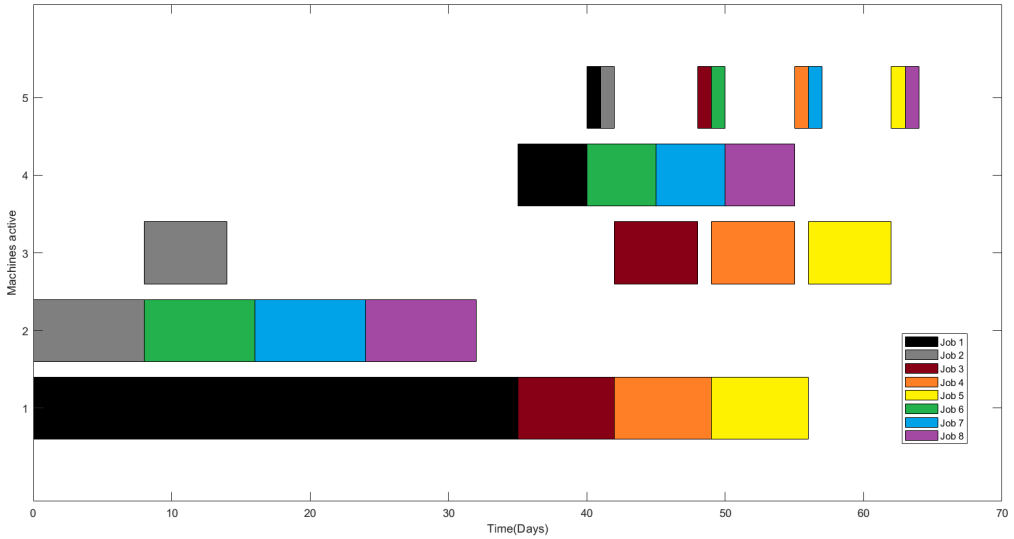

Figure 5-28: Initial schedule with fixed order and routes but updated process time $p_1(k = 1) = 35$.

Some interesting things can be observed in the figures. First of all, note that the at time instant $\kappa = 1$. already some processes are fixed, but not necessarily modes. In other words, if we look at both figures, job 1 and job 2, are already fixed for machine 1 and machine 2 since the starting time is at $\kappa = 0$ which is less than $\kappa = 1$. However, when looking at the black block of job 1, it can be observed that in figure 5-28 it follows mode/route 2, where in figure 5-29, it follows mode/route 1. This if of course caused by the sake that machine 4 has a lower

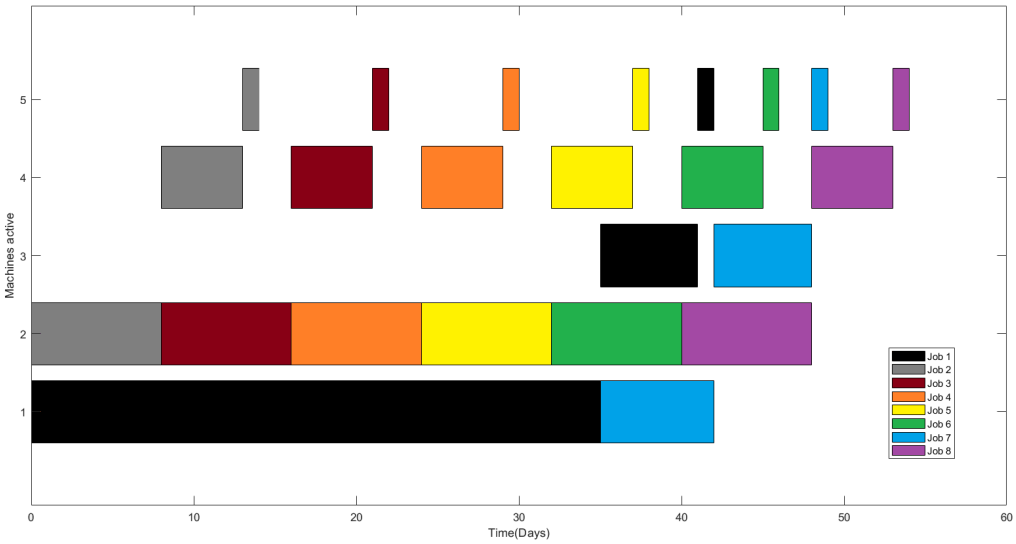

Figure 5-29: Updated schedule with process time $p_1(k = 1) = 35$.

processing time, than machine 3, i.e. $p_3(k) = 6$ and $p_4(k) = 5$. By changing from mode 2 to mode 1 for job 1, it allows the other jobs to go through machine 4 instead of machine 3. Note that this is exactly what is not possible in the cycle-based updated scheduler since the mode would be fixed such that job 1 processed in machine 4 is also fixed.

Another interesting observation, is of course the total amount of jobs which go through machine 1, and the total amount of jobs which go through machine 2. This is decreased from the initial schedule with respect to the final schedule, with a amount of 4 to 2 in machine 1. The amount of jobs in machine 2 increased from 4 to 6. This is of course logical, since the process time of job 1 increased drastically. The mode changes are summarized in table 5-32. In the table it can be better observed, the mode/route for job 1, $\ell(1)$ changes from 2 to 1,

	$\ell(1)$	$\ell(2)$	$\ell(3)$	$\ell(4)$	$\ell(5)$	$\ell(6)$	$\ell(7)$	$\ell(8)$
Routes at initial schedule $\kappa = 0$	2	3	1	1	1	4	4	4
Routes at updated schedule $\kappa = \kappa_{final}$	1	4	4	4	4	4	1	4

Table 5-32: Modes changes of the initial schedule with respect to the final schedule.

while the time is already passed the starting time from the mode/route. In the cyclic update case, this would not be possible. The mode would be fixed because the update is regarding to the cycle k. The update value of the sum of the endtimes, the cost function, decreased drastically if the intial schedule and order with updated process times, is compared with the updated schedule and order with updated process times. The values of the sum of the end times are $\sum_{end} = 414$ and $\sum_{end} = 287$ for respectively the initial schedule and the final schedule. It can be concluced therefore, that the update with respect to certain disruption is very beneficial, but this is of course logical since at the initial schedule all jobs are ordered behind job 1, such that if job 1 has a very large delay all jobs need to wait for job 1 instead of reorder. This could also be observed in the cyclic-updated scheduler.

Disturbances - Machine M_1 and M_2

For the second approach only machine 1 and 2 deviate in processing times. One example of a realization is given, in which several characteristic behaviours of the adapted scheduler are outlined. To do this, a lot of tables are shown for the sake of understanding what is happening inside the scheduler. One could imagine, when the last job is finished at time-instant $\kappa = 50$ and taking a time-step of $\kappa = 1$, 49 time steps are obtained with all updates of process times and modes. It is rather hard to show this whole process, since the report would become massive and cluttered. Still it is very worthy to look at some of the values in the tables. Afterwards, several other realizations are obtained, such that the overal behaviour of the updated scheduler can be judged.

First of all in tables 5-33 and 5-34 it can be brilliantly observed what the predictions

	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$
$p_1(k=1)$	**7.00**	**7.60**	**6.90**	**6.50**	**6.70**	**6.70**	**6.60**	**6.60**	6.60	6.60	6.60	6.60	6.60	6.60
$p_1(k=2)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=3)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=4)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	**6.70**	**6.60**	**6.80**	**6.60**	**6.50**	**6.30**	**6.20**
$p_1(k=5)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	**7.10**
$p_1(k=6)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=7)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=8)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00

Table 5-33: Updated process times for machine 1 per discrete time-instant κ when only machine 1 and 2 have deviating processing times.

	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$
$p_2(k=1)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=2)$	**8.00**	**8.00**	**8.40**	**8.50**	**8.90**	**8.70**	**9.30**	**9.60**	**9.50**	**9.60**	**9.50**	9.50	9.50	9.50
$p_2(k=3)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	**8.30**	**8.80**	**9.10**	**9.90**
$p_2(k=4)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=5)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=6)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=7)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=8)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00

Table 5-34: Updated process times for machine 2 per discrete time-instant κ when only machine 1 and 2 have deviating processing times.

are doing. The numbers in bold are active machines where the predictions can be considered "online" per discrete time-step κ. When a specific job has not yet been started at discrete time-instant κ, the estimated end-time is of course equal to the nominal end-time which for machine 1 is equal to $p_1 = 7$ and for machine 2 equal to $p_2 = 8$. This can be noticed when looking at column $\kappa = 1$ in both tables where only job $k = 1$ and $k = 2$ are started and therefore the expected end-time for other jobs if the routes choose to go through machine 1 or 2, are equal to $p_1 = 7$ and $p_2 = 8$. If we now take for instance job 1, which is taking route 2(through machine 1) as can be observed in table 5-35, we can see the prediction of the end-time is getting more accurate when time increases, these are the numbers in bold of row 1 in table 5-33. When the discrete time-instant $\kappa = 7$, we can observe in the specific column of $\kappa = 7$, process 1 for job 1 is finished after $p_1(k = 1) = 6.60$ and job 4 starts in machine 1 as can be observed by the sequence of bold numbers in the column of $\kappa = 7$ in 5-33. After

discrete time-instant $\kappa = 7$, process 1 in machine 1 for job 1 is finished, and therefore the end-time of process 1 of job 1 in machine 1 is fixed and therefore equal to $p_1(1) = 6.60$ for all future discrete time-instants $\kappa = 7, ..., \kappa_{final}$. Note that $\kappa = 13$ is not the final discrete time-step for the whole simulation, but is taken as a final discrete time-step for the table to substantiate and explain specific behavioural aspects of the dynamical scheduler. If we now

	$\ell(k=1)$	$\ell(k=2)$	$\ell(k=3)$	$\ell(k=4)$	$\ell(k=5)$	$\ell(k=6)$	$\ell(k=7)$	$\ell(k=8)$
$\kappa = 0$	2	3	1	1	1	4	4	4
$\kappa = 1$	2	3	2	2	3	3	2	3
$\kappa = 2$	2	3	2	3	2	3	2	4
$\kappa = 3$	2	3	3	2	2	3	2	4
$\kappa = 4$	2	3	3	2	2	3	2	4
$\kappa = 5$	2	3	3	2	2	3	2	4
$\kappa = 6$	2	3	3	2	2	3	2	4
$\kappa = 7$	2	3	3	2	2	3	2	4
$\kappa = 8$	2	3	3	2	2	3	2	4
$\kappa = 9$	2	3	3	2	2	3	2	4
$\kappa = \mathbf{10}$	**2**	**3**	**3**	**2**	**2**	**4**	**1**	**4**
$\kappa = \mathbf{11}$	**2**	**3**	**3**	**2**	**2**	**4**	**1**	**2**
$\kappa = 12$	2	3	3	2	2	4	1	2
$\kappa = 13$	2	3	3	2	2	3	2	2
$\vdots$	$\vdots$							$\vdots$
$\kappa = \kappa_{final}$	2	3	3	2	2	4	1	2

Table 5-35: Updated modes/routes for all jobs $k, .., k + N_p$ per discrete time-instant κ when machines 1 and 2 have deviating processing times.

look at the updated modes per discrete time-instant κ in table 5-35, we can observe something remarkable. If you look closer to the modes chosen per discrete time-step κ, you can observe the modes chosen are always equally assigned to machine 1 and machine 2, such that for 8 jobs, 4 jobs will go through machine 1 and 4 jobs will go through machine 2. For instance at $\kappa = 0$, route 1 is chosen 3 times, route 2 is chosen once, route 3 is chosen once and route 4 is chosen 3 times. Since routes 1 and 2 go through machine 1, and routes 3 and 4 go through machine 2, an easy calculation of the sums gives that 4 routes go through machine 1 and 4 jobs go through machine 2. This is logical if you make a simple calculation of the end-times of the processes in respectively machine 1 and 2, when taking the nominal process times $p_1 = 7$ and $p_2 = 8$. If they follow each other directly namely, the endtimes in machine 1 will equal $7, 14, 21, 28$ and in machine 2 will equal $8, 16, 24, 32$. Now if we sum up these endtimes and compare these results with 5 jobs going through either machine 1 or 2, and 3 through either machine 1 or 2, we get:

$$\begin{aligned} \textbf{Case 1} &= (7 + 14 + 21 + 28) + (8 + 16 + 24 + 32) = 150 \\ \textbf{Case 2} &= (7 + 14 + 21) + (8 + 16 + 24 + 32 + 40) = 162 \\ \textbf{Case 3} &= (7 + 14 + 21 + 28 + 35) + (8 + 16 + 24) = 153 \end{aligned} \tag{5-6}$$

Where it follows directly, with nominal process times, case 1 must be chosen. However if we go back to table 5-35, and observe the rows for discrete-time instants $\kappa = 10$ and $\kappa = 11$, we can see that the total amount of jobs going through machine 1 and machine 2 for $\kappa = 10$ is respectively 4 and 4, while the total amount of jobs going through machine 1 and machine 2

for $\kappa = 11$ is respectively 5 and 3. This is caused by the last job which switches from route 4 to route 2. This can be substantiated when looking at the predicted or passed process times from tables 5-33 and 5-34. If we make the same calculation as in 5-6 namely with updated/predicted process times for machine 1 and 2 up till discrete time-instant $\kappa = 11$ and take nominal process times for the processes in machine 1 and 2 which start in future discrete time-instances, we get:

$$\begin{aligned}\textbf{Case 1} = &\underbrace{(6.60 + (6.60 + 6.50) + (6.60 + 6.50 + 7) + (6.60 + 6.50 + 7 + 7))}_{\text{4 jobs through machine 1}} \\ &+ \underbrace{(9.50 + (9.50 + 8.80) + (9.50 + 8.80 + 8) + (9.50 + 8.80 + 8 + 8)}_{\text{4 jobs through machine 2}} = 155.30 \\ \textbf{Case 2} = &\ (6.60 + (6.60 + 6.50) + (6.60 + 6.50 + 7) \\ &\underbrace{+(6.60 + 6.50 + 7 + 7) + (6.60 + 6.50 + 7 + 7 + 7))}_{\text{5 jobs through machine 1}} \\ &+ \underbrace{(9.50 + (9.50 + 8.80) + (9.50 + 8.80 + 8))}_{\text{3 jobs through machine 2}} = 155.10\end{aligned} \tag{5-7}$$

What can be observed is that the case 1 where we take 4 jobs through machine 1 and 4 jobs through machine 2 has a higher sum of endtimes than when taking 5 jobs through machine 1 and 3 jobs through machine 2. This is the reason why the scheduler at the particular time-instant switches the modes. This can be better observed in a GANTT-chart as is shown in figure 5-30 and 5-31. It can be brilliantly observed that job 8, the purple block, switches from

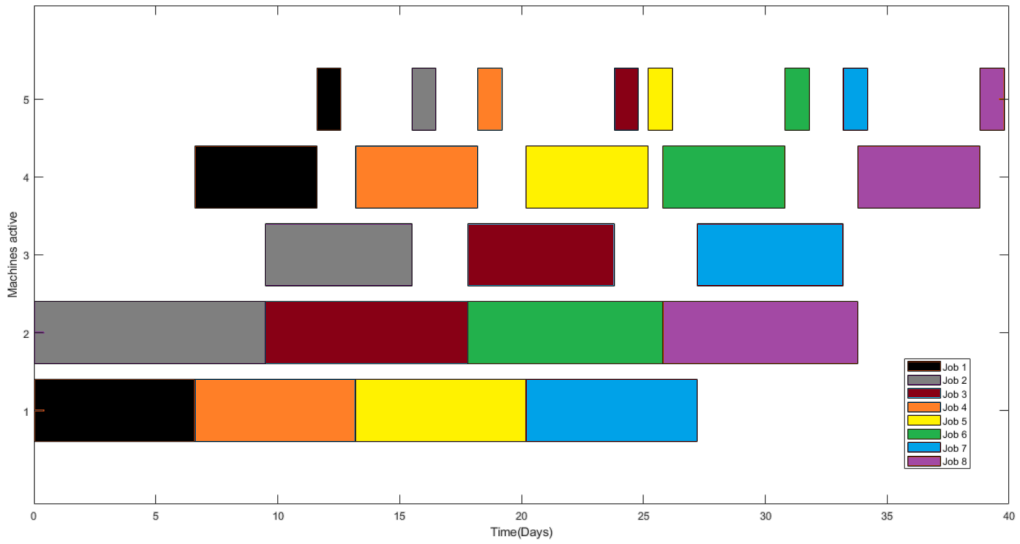

Figure 5-30: GANTT-chart of the schedule at $\kappa = 10$ with the updated processing times up till $\kappa = 10$.

mode 4 to mode 2 by changing the initial process from machine 2 to machine 1. Furthermore, it can be shown that job 3 and 5, the red and yellow blocks are closer together in machine 5, due to the fact the prediction of job 3 in machine 2 increased from $p_1(3) = 8.30$ at discrete time-instant $\kappa = 10$ to $p_1(3) = 8.80$ at discrete time-instant $\kappa = 11$. This can also be observed when looking at job 6 and 7, the green and lightblue blocks, which are closer together

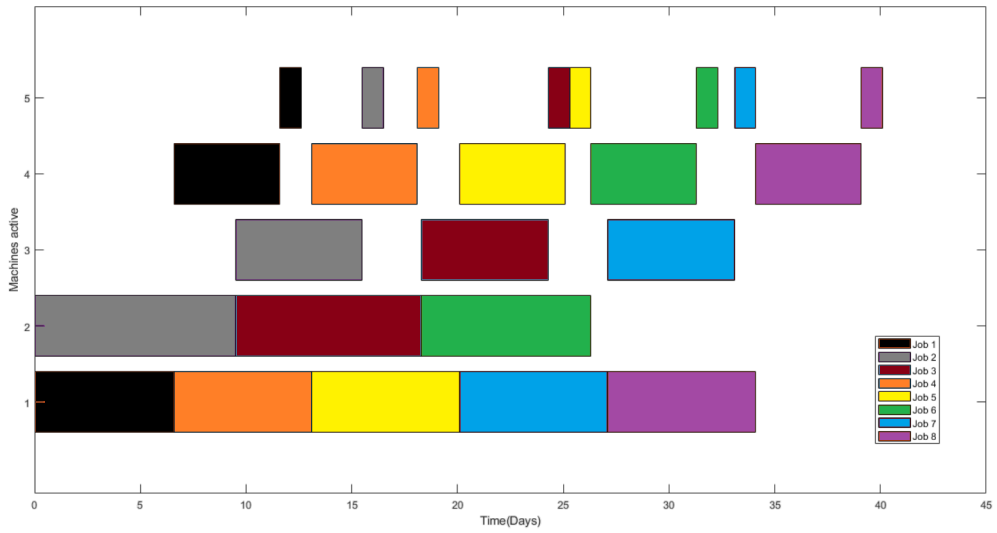

Figure 5-31: GANTT-chart of the schedule at $\kappa = 11$ with the updated processing times up till $\kappa = 11$.

in machine 5, due to the fact job 6, the green block, needs to wait at job 3, the red block, in machine 2. The value of the cost function decreased by this decision from FVAL = 199.70 to FVAL = 198.30.

When we now go a few discrete time-instances further, at $\kappa = 19$, the process-time of $p_2(k = 3)$ became even larger, $p_2(k = 3) = 10$. The result is that job 3, the red block, is probably causing a delay for job 5, the yellow block in machine 5, as can already be concluded in figure 5-31 that they are precisely merged together. In the next GANTT-chart 5-32, it can be observed the update at $\kappa = 19$ the order of the yellow block, job 5, is placed in front of the red block, job 3. Moreover, the red block is taking so long, the green and lightblue block, respectively job 6 and 7, not only changed routes, but also the order in machine 5 as can be observed when comparing 5-31 and 5-32. These given GANTT-charts as well as the tables give a nice insight in the adaptive behaviour of the scheduler, where it updates every discrete time-instant κ.

Ultimately, the initial schedule with fixed order and routes but updated process times is compared with the final scheduler with updated order, routes and process times. First the results for the ultimate process times are given, where after the results are compared via the active machines. This is done the same as for the approaches in the cycle-updated cases. The results are shown in tables 5-36, 5-37 and 5-38. Now if we have the same procedure already done in the previous paragraphs $\sum_{total} = -3.50 + 2.90 = -0.60$ and $\sum_{total} = -1.80 + 0.90 = -0.90$ for respectively the final schedule and the initial schedule we can obtain $\sum_{dif} = -0.60 - (-0.90) = +0.30$. Filling in these values in the cost function values, the results are shown in 5-39. Where a beneficial value of the cost function is obtained by updating the schedule every discrete time-instance κ. Now repeating this step for 6 realizations will give the results shown in table 5-40. The results are fine, but still marginal. A decrease can observed when looking in the FVAL_{new} column. Extending this to a monte-carlo simulation with $n = 100$ gives the result that a certain decrease can be observed. It is

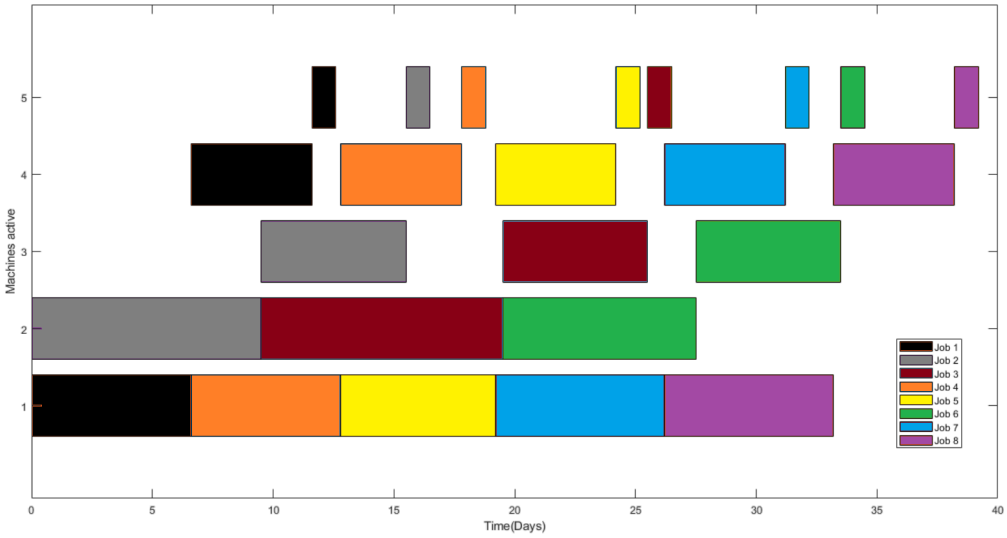

Figure 5-32: GANTT-chart of the schedule at $\kappa = 19$ with the updated processing times up till $\kappa = 19$.

$p_1(k)$	$p_2(k)$
6.60	8.00
7.00	9.50
7.00	10.00
6.20	8.00
6.40	8.00
7.00	7.40
6.70	8.00
5.60	8.00

Table 5-36: Ultimate updated process times for the processes for 1 and 2 where only machine 1 and 2 are subjected to deviations with respect to the discrete time-counter κ.

	$p_1(k) - p_1$	$p_2(k) - p_2$
$k = 1$	**-0.40**	0
$k = 2$	0	**1.50**
$k = 3$	0	**2.00**
$k = 4$	**-0.80**	0
$k = 5$	**-0.60**	0
$k = 6$	0	**-0.60**
$k = 7$	**-0.30**	0
$k = 8$	**-1.40**	0
$\sum$	-3.50	+2.90

Table 5-37: Differences between the updated ultimate process times with respect to the nominal process times. In bold the processes taken by the final scheduler.

	$p_1(k) - p_1$	$p_2(k) - p_2$
$k = 1$	**-0.40**	0
$k = 2$	0	**1.50**
$k = 3$	**0**	2.00
$k = 4$	**-0.80**	0
$k = 5$	**-0.60**	0
$k = 6$	0	**-0.60**
$k = 7$	-0.30	**0**
$k = 8$	-1.40	**0**
$\sum$	-1.80	+0.90

Table 5-38: Differences between the updated ultimate process times with respect to the nominal process times. In bold the processes taken by the initial scheduler.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule	199.90	+0.30	200.20
Updated schedule	194.20	+0.30	194.20

Table 5-39: Comparison of the shown realization of the sum of the endtimes between the initial schedule with fixed route and order and updated process times, and the final schedule with updated schedule.

	FVAL	$\sum$	FVAL_{new}
Initial schedule(1)	209.90	-1.10	208.80
Updated schedule(1)	204.50	-1.10	204.50
Initial schedule(2)	198.90	-1.10	197.80
Updated schedule(2)	196.50	-1.10	196.50
Initial schedule(3)	209.10	+0.20	209.30
Updated schedule(3)	203.10	+0.20	203.10
Initial schedule(4)	198.00	-0.40	197.60
Updated schedule(4)	191.20	-0.40	191.20
Initial schedule(5)	203.10	+0.00	203.10
Updated schedule(5)	199.90	+0.00	199.90
Initial schedule(6)	196.90	-1.00	195.90
Updated schedule(6)	191.40	-1.00	191.40

Table 5-40: Comparison of 6 realizations of the sum of the endtimes between the schedule with fixed route and order but updated process times, and the final schedule with updated process times.

marginal but it can be observed a decrease of -1.93% can be obtained in table 5-41.

$\text{FVAL}_{new} - FinalScheduler$	$\text{FVAL}_{new} - Initialscheduler$	$\sum_{dif}$	Decrease in cost(%)
193.13	197.36	-0.20	-1.93%

Table 5-41: Montecarlo simulation of the time-based updated with $n = 100$ with average value of the final scheduler its cost, average value of the initial scheduler its cost, average $\sum_{dif}$ and average decrease in total endtime with respect to the initial schedule in % for deviating machines 1 and 2.

Disturbances - Machine M_3 and M_4

In this paragraph an approach is used where the processing times for machine 3 and 4 are subjected to deviations. First an illustrative realization is outlined, where characteristics and behavioural aspects of the adaptive scheduler can be analyzed using several tables and figures. Whereafter multiple realizations are obtained, to see the beneficial value of the adaptive scheduler.

First of all an example realization is obtained, where certain switches can be discussed, or concluded by the means of varying processing times. To begin, it is nice to see the tables where it can be seen, live updated prediction are obtained. Since the processing times for machine 1 and 2 are assumed to be constant $p_1 = 7$ and $p_2 = 8$, the deviating processes p_3 and p_4 start the fastest after respectively 7 or 8. Therefore in the following tables for the first jobs, the discrete time range is taken from $\kappa = 7$ up till $\kappa = 20$. As can be concluded,

	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$	$\kappa=14$	$\kappa=15$	$\kappa=16$	$\kappa=17$	$\kappa=18$	$\kappa=19$	$\kappa=20$
$p_3(k=1)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=2)$	6.00	6.00	**6.70**	**7.20**	**7.80**	**8.40**	**8.90**	**8.50**	**8.40**	**8.20**	**8.20**	8.20	8.20	8.20
$p_3(k=3)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=4)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=5)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=6)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=7)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=8)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00

Table 5-42: Updated process times for machine 3 per discrete time-instant κ when only machine 3 and 4 have deviating processing times.

	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$	$\kappa=14$	$\kappa=15$	$\kappa=16$	$\kappa=17$	$\kappa=18$	$\kappa=19$	$\kappa=20$
$p_4(k=1)$	5.00	**5.40**	**6.30**	**6.60**	**6.50**	**6.40**	**6.50**	**6.50**	6.50	6.50	6.50	6.50	6.50	6.50
$p_4(k=2)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=3)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	**4.50**	**4.30**	**4.30**	**4.10**	**4.10**	4.10
$p_4(k=4)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	**4.80**	**4.60**
$p_4(k=5)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=6)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=7)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=8)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00

Table 5-43: Updated process times for machine 4 per discrete time-instant κ when only machine 3 and 4 have deviating processing times.

the machines can become idle. Looking at table 5-42 for example after discrete time-instant $\kappa = 17$, the machine does not work for a while. This is logical, since the first 2 processes in machine 1 and machine 2 have longer nominal processing times than the processes in machine

3 and machine 4. Since the processes in machine 3 and 4 therefore have to wait, certain gaps can occur, where machines 3 and 4 become idle. This can be better observed when looking at figure 5-33, where spaces between the jobs in machine 3 and 4 occur. One could ask themselves, if the machines are already idle, which beneficial effect could occur. Well, not only the machines could take longer or shorter in machines 3 and 4, with machines 1 and 2 delivering constant processing times, ordering in machine 5 could also occur with a result of deviating processing times in machine 3 and 4. Also machines 3 and 4 could take longer or shorter, where they both become idle, and the machine with the shortest amount of processing time, can do one extra job, in this case machine 4, since it has the lowest processing time. This can be better observed by the mode changing of the realization described already a little bit in tables 5-42 and 5-43, but is summarized in table 5-44. The mode change sequences in bold

	$\ell(k=1)$	$\ell(k=2)$	$\ell(k=3)$	$\ell(k=4)$	$\ell(k=5)$	$\ell(k=6)$	$\ell(k=7)$	$\ell(k=8)$
$\kappa=7$	2	3	1	4	1	4	4	1
$\kappa=8$	2	3	1	4	1	4	4	1
$\kappa=\mathbf{9}$	**2**	**3**	**2**	**3**	**2**	**3**	**2**	**3**
$\kappa=\mathbf{10}$	**2**	**3**	**2**	**3**	**2**	**3**	**2**	**4**
$\kappa=11$	2	3	1	4	2	3	2	4
$\kappa=12$	2	3	2	4	1	4	1	4
$\kappa=13$	2	3	2	4	1	4	1	4
$\kappa=14$	2	3	2	4	1	4	1	4
$\kappa=15$	2	3	2	4	1	4	1	4
$\kappa=16$	2	3	2	4	1	4	1	4
$\kappa=17$	2	3	2	4	1	4	1	4
$\kappa=18$	2	3	2	4	1	4	1	4
$\kappa=19$	2	3	2	4	1	4	1	4
$\kappa=20$	2	3	2	4	1	4	1	4
$\vdots$	$\vdots$							$\vdots$
$\kappa=\kappa_{final}$	2	3	2	4	1	4	1	4

Table 5-44: Updated modes/routes for all jobs $k,..,k+N_p$ per discrete time-instant κ when machines 3 and 4 have deviating processing times.

for discrete time-instances $\kappa=9$ and $\kappa=10$ have 1 remarkable difference. Not to directly see on the eye, but when looking close, it can be observed the modes changed from equally 4 jobs in both machine 3 and machine 4 individually, to 3 jobs in machine 3 and 5 jobs in machine 4. To conclude, the GANTT-charts are shown with processing times according to the process times per discrete time-instance $\kappa=9$ and $\kappa=10$. It can be concluded only the purple block is shifted from machine 3 at discrete time-instant $\kappa=9$ to machine 4 at discrete time-instant $\kappa=10$. For now it has no reason to do so, since the processes in machine 1 and 2 are fixed, and the processes in machine 3 and 4 have open spaces inbetween them, which means they are idle and can be used. However, one could imagine if job 7, the lightblue block, would finish its job faster in machine 4, such that the end of job 8 in machine 2, the purple block, would align perfectly with the light blue block in machine 4, it could take machine 4. Since machine 4 has nominal process time $p_4=5$ which is lesser than $p_3=6$, all jobs would take machine 4 instead of machine 3 if possible. The same could be said when for instance job 8, the purple block, would have a very large delay in machine 2, such that both machines 3 and 4 are idle by the end of job 8 in machine 2. In the initial schedule it would still use machine 3, however when this situation occurs it could take machine 4 as well since it became idle, with a beneficial time gain of 1 time-unit.

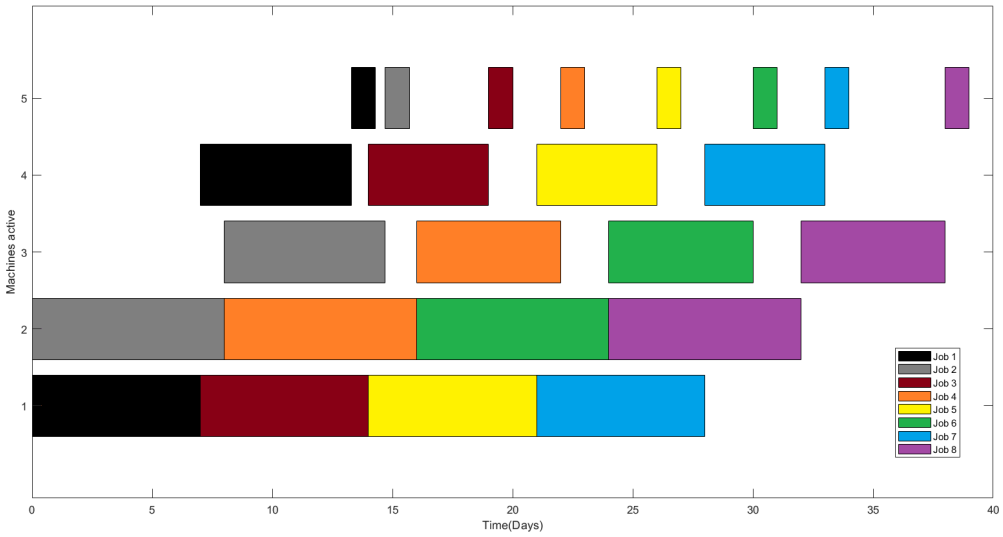

Figure 5-33: GANTT-chart of the jobs of the schedule at $\kappa = 9$.

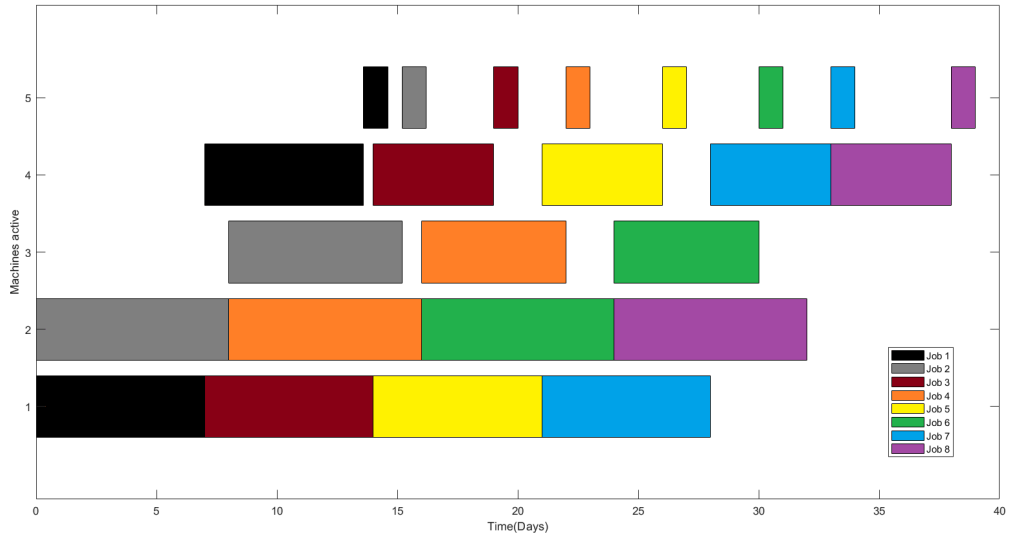

Figure 5-34: GANTT-chart of the jobs of the schedule at $kappa = 10$.

For now to conclude, machine 3 and 4 could have a beneficial effect when both machines become idle due to processing times in machine 1 and 2 take longer, or processes in machine 3 or 4 take shorter or longer, but can also contribute to the order of processes in machine 5. The results of the above mode switching and GANTT-charts can be observed by the updated modes at κ_{final} in table 5-44. It can be seen at last, 5 jobs go through machine 4, where 3 jobs go through amchine 3. The beneficial effects are summarized in tables 5-44, 5-46, 5-47 and 5-48.

Now by repeating the steps already done a few times now, we can come up with $\sum_{total} = +3.20 + 0.30 = +3.50$ and $\sum_{total} = +1.90 + 1.20 = +3.10$ such that the difference becomes $\sum_{dif} = +3.50 - 3.10 = +0.40$. It is shown in 5-48 the adjusted scheduling has a beneficial

$p_3(k)$	$p_4(k)$
6.00	6.50
8.20	5.00
6.00	4.10
6.00	4.40
5.70	5.00
6.00	5.00
7.30	5.00
6.00	5.30

Table 5-45: Ultimate updated process times for the processes for 3 and 4 where only machine 3 and 4 are subjected to deviations with respect to the discrete time-counter κ.

	$p_3(k) - p_3$	$p_4(k) - p_4$
$k = 1$	0	**1.50**
$k = 2$	**2.20**	0
$k = 3$	0	**-0.90**
$k = 4$	0	**--0.60**
$k = 5$	**-0.30**	0
$k = 6$	0	**0**
$k = 7$	**1.30**	0
$k = 8$	0	**0.30**
$\sum$	+3.20	+0.30

Table 5-46: Differences between the updated ultimate process times with respect to the nominal process times. In bold the processes taken by the final scheduler.

	$p_3(k) - p_3$	$p_4(k) - p_4$
$k = 1$	0	**1.50**
$k = 2$	**2.20**	0
$k = 3$	**0**	-0.90
$k = 4$	0	**--0.60**
$k = 5$	**-0.30**	0
$k = 6$	0	**0**
$k = 7$	1.30	**0**
$k = 8$	**0**	0.30
$\sum$	+1.90	+0.90

Table 5-47: Differences between the updated ultimate process times with respect to the nominal process times. In bold the processes taken by the initial scheduler.

effect on the specific realization with almost 6 time-units. Of course, the above steps are just to illustrate certain behavioural aspect of the switching. Again, to conclude carefully, 6 realizations are obtained. In the following realizations in table 5-49, the same procedure is done to compare the initial and final schedule with respect to the routing and ordering with updated processing times. As can be concluded, the change of modes or orders has some beneficial effect on the value of the cost function, the sum of the end times of the jobs.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule	203.70	+0.40	204.10
Updated schedule	198.60	+0.70	198.60

Table 5-48: Comparison of the shown realization of the sum of the endtimes between the initial schedule with fixed route and order but updated process times, and the final schedule with updated schedule.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule(1)	198.00	-0.20	197.80
Updated schedule(1)	195.50	-0.20	195.50
Initial schedule(2)	200.00	+2.10	200.00
Updated schedule(2)	196.30	+2.10	196.30
Initial schedule(3)	195.70	-1.20	194.50
Updated schedule(3)	193.40	-1.20	193.40
Initial schedule(4)	198.90	-2.00	196.90
Updated schedule(4)	192.50	-2.00	192.50
Initial schedule(5)	201.70	+1.70	201.70
Updated schedule(5)	194.60	+1.70	194.60
Initial schedule(6)	199.30	+3.80	203.10
Updated schedule(6)	201.00	+3.80	201.00

Table 5-49: Comparison of 6 realizations of the sum of the endtimes between the schedule with fixed route and order but updated process times, and the final schedule with updated process times.

Therefore a small, but significant beneficial effect can be obtained by the updated schedule with respect to the deviating processing times of machine 3 and 4. Extending this to a montecarlo simulation of $n = 100$ substantiates this in table 5-50.

$\text{FVAL}_{new} - FinalScheduler$	$\text{FVAL}_{new} - Initialscheduler$	$\sum_{dif}$	Decrease in cost(%)
195.09	196.96	+0.35	-0.80%

Table 5-50: Montecarlo simulation of the time-based updated with $n = 100$ with average value of the final scheduler its cost, average value of the initial scheduler its cost, average $\sum_{dif}$ and average decrease in total endtime with respect to the initial schedule in % for deviating processing times of machines 3 and 4.

Disturbances - All machines

In this last paragraph, all things come together. The updated scheduler with respect to each discrete time-instant κ with all machines having updated processing times. It is however, very hard to illustrate or conclude something by mean of behaviour of the switching modes, since all process times deviate all the time. Therefore, first some tables and figures are obtained to show the adaptive behaviour of the discrete time updated scheduler, whereafter the results are more shown in a numerical way with respect to the value of the cost function.

First the live predictions are shown with respect to the constantly updating processing times for all the machinens with respect to each discrete time-couner κ. It is nicely illustrated in the tables 5-51, 5-52, 5-53, 5-54 and 5-55. The numbers in black bold are for job 1, the numbers in gray bold are for job 2, the number in red bold are for job 3 and the numbers in orange bold are for job 4. First of all it can be very nicely observed how the prediction take care per discrete time-step κ. The numbers evolve per discrete time-step for a better prediction and when arriving at discrete time-instant where it is finished, it can be observerd the job afterwards in that machine starts. At least, for machine 1 and 2 this is the case as can be shown in 5-51 and 5-52 where the red bold numbers follow up the black bold numbers at $\kappa = 10$ and the orange bold numbers follow up the gray bold numbers at $\kappa = 8$. Furthermore, the process after process 1 or 2 in machine 1 or 2 is also nicely shown. At $\kappa = 10$ in table 5-51 it can be observed in 5-53 the process in machine 3 starts at $\kappa = 10$ and already a prediction is given. The same can be told for the gray bold number with respect to tables 5-52 and 5-54 at time-instant $\kappa = 8$.

	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$
$p_1(\mathbf{k=1})$	**7.00**	**7.40**	**6.90**	**7.70**	**8.90**	**9.10**	**8.90**	**8.90**	**9.30**	**9.30**	**9.30**	9.30	9.30	9.30
$p_1(k=2)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(\mathbf{k=3})$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	**7.60**	**7.50**	**7.90**	**8.20**
$p_1(k=4)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=5)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=6)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=7)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
$p_1(k=8)$	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00

Table 5-51: Updated process times for machine 1 per discrete time-instant κ when all machines have deviating processing times. The different jobs k are coloured in different colours.

	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$
$p_2(k=1)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(\mathbf{k=2})$	**8.00**	**7.50**	**7.80**	**8.10**	**8.00**	**7.70**	**7.50**	**7.50**	**7.40**	7.40	7.40	7.40	7.40	7.40
$p_2(k=3)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(\mathbf{k=4})$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	**7.80**	**7.80**	**8.00**	**7.90**	**8.20**	**8.10**
$p_2(k=5)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=6)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=7)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
$p_2(k=8)$	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00

Table 5-52: Updated process times for machine 2 per discrete time-instant κ when all machines have deviating processing times. The different jobs k are coloured in different colours.

When we now look closer to 5-55, we can observe some special things. First of all it can be observed the modes for $\ell(k=1)$ and $\ell(k=2)$ change while the starting time of the first pro-

	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$
$p_3(\mathbf{k=1})$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	**6.20**	**6.60**	**7.50**	**6.50**
$p_3(k=2)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=3)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=4)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=5)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=6)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=7)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
$p_3(k=8)$	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00

Table 5-53: Updated process times for machine 3 per discrete time-instant κ when all machines have deviating processing times. The different jobs k are coloured in different colours.

	$\kappa=0$	$\kappa=1$	$\kappa=2$	$\kappa=3$	$\kappa=4$	$\kappa=5$	$\kappa=6$	$\kappa=7$	$\kappa=8$	$\kappa=9$	$\kappa=10$	$\kappa=11$	$\kappa=12$	$\kappa=13$
$p_4(k=1)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(\mathrm{k=2})$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	**5.80**	**6.60**	**7.50**	**6.90**	**6.60**	**6.60**
$p_4(k=3)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=4)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=5)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=6)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=7)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
$p_4(k=8)$	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00

Table 5-54: Updated process times for machine 4 per discrete time-instant κ when all machines have deviating processing times. The different jobs k are coloured in different colours.

	$\ell(\mathbf{k=1})$	$\ell(\mathrm{k=2})$	$\ell(\mathbf{k=3})$	$\ell(\mathbf{k=4})$	$\ell(k=5)$	$\ell(k=6)$	$\ell(k=7)$	$\ell(k=8)$
$\kappa=0$	**2**	3	**2**	**3**	2	3	3	2
$\kappa=1$	**2**	3	**2**	**3**	2	3	3	2
$\kappa=2$	**2**	3	**2**	**3**	2	3	3	2
$\kappa=3$	**2**	3	**2**	**3**	2	3	3	2
$\kappa=4$	**1**	4	**2**	**3**	2	3	3	2
$\kappa=5$	**1**	4	**4**	**1**	2	3	3	2
$\kappa=6$	**1**	4	**1**	**4**	2	3	3	2
$\kappa=7$	**1**	4	**1**	**4**	2	3	2	3
$\kappa=8$	**1**	4	**1**	**4**	2	3	2	3
$\kappa=9$	**1**	4	**1**	**4**	1	4	2	3
$\kappa=10$	**1**	4	**1**	**4**	4	1	2	3
$\kappa=11$	**1**	4	**1**	**4**	4	1	2	3
$\kappa=12$	**1**	4	**1**	**4**	4	1	2	3
$\kappa=13$	**1**	4	**1**	**4**	4	1	2	3
$\vdots$	$\vdots$							$\vdots$
$\kappa=\kappa_{final}$	**1**	4	**1**	**4**	4	1	1	4

Table 5-55: Updated modes/routes for all jobs $k,..,k+N_p$ per discrete time-instant κ when machines 1,2,3 and 4 have deviating processing times.

cesses of these specific jobs already started, i.e. they both start at $\kappa=0$ as can be observed in 5-51 and 5-52. This is however possible since the changes are from mode $2\rightarrow1$ and $3\rightarrow4$. These changes are only changing the routes regarding machine 3 and 4, and not regarding machine 1 and 2, which is not possible since these processes already started. Note that this is one of the major differences with respect to the cylic updated case, where very cycle in the future, the modes for the cycles in the passed are fixed. However, if we now look at the mode changes for $\ell(k=3)$ and $\ell(k=4)$ at discrete time-instances $\kappa=4$ and $\kappa=5$, we can observe the modes change from $2\rightarrow4$ and $3\rightarrow1$. These changes are in fact regarding machine 1 and

2. However, as can be observed in tables 5-51 and 5-52, the endtime of these processes from the previous jobs are at $\kappa = 10$ and $\kappa = 8$ where $\kappa = 4$ and $\kappa = 5$ are time-instances before the endtime of the processes from the previous jobs. This makes it possible to make the specific mode change. When going further into the future, i.e. κ increases, it can be observed more modes get fixed in table 5-55, or can only make changes from $1 \rightarrow 2$ and $3 \rightarrow 4$ or vice versa, since these mode changes take place in machine 3 and 4 where the processes may not have been started yet.

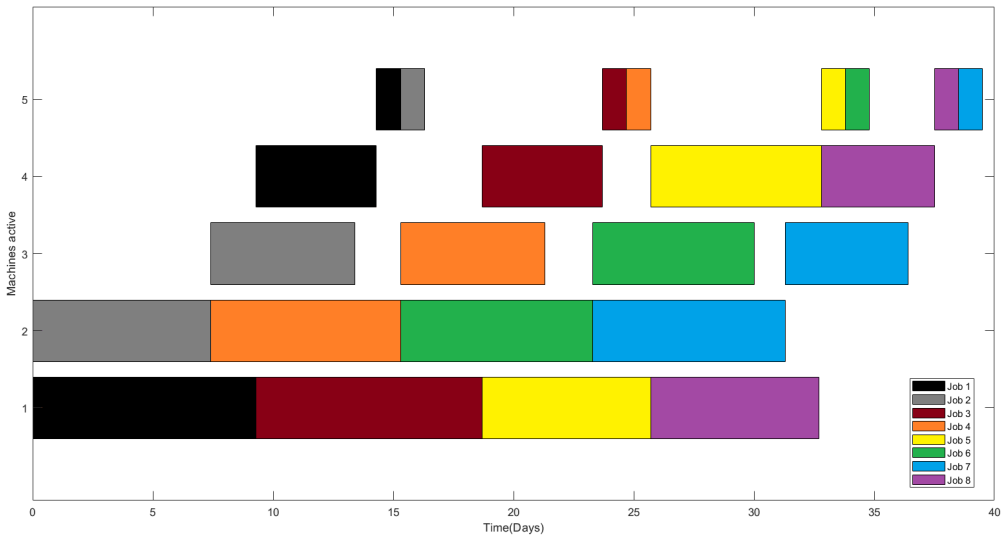

Figure 5-35: GANTT-chart of the initial schedule with fixed order and route but updated processing times.

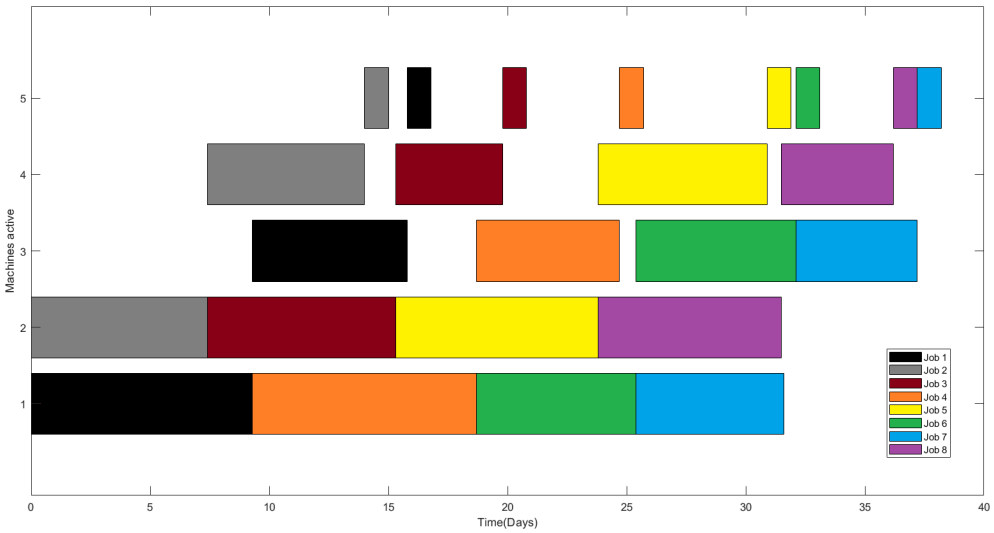

Figure 5-36: GANTT-chart of the final updated schedule with updated process times.

The GANTT-charts in 5-35 and 5-36 show nicely how the initial schedule with fixed order

and routes but updated process times deviate from the final schedule with updated process times. First of all the black block, job 1, takes a lot longer than the nominal time $p_1 = 7$ as can also be concluded when following the predictions in 5-51. The gray block however, job 2, has a small decrease from its nominal process time $p_2 = 8$. What can be observed is that the predictions the system gets at $\kappa = 3$ in tables 5-51 and 5-52 give enough reason to change the modes as can be observed in rows $\kappa = 3$ and $\kappa = 4$ in table 5-55. This is for the benefit of job 2 being ordered in front of job 1 in machine 5, and therefore decreases the value of the cost function, the sum of end-times of all the jobs.

To conclude anything about the benefits of these mode changes, the initial schedule is compared with the final schedule, as is already done is all the paragraphs. The next tables 5-56, 5-57, 5-58 and 5-59 are meant to illustrate this. Now the following conclusions can be

$p_1(k)$	$p_2(k)$	$p_3(k)$	$p_4(k)$
9.30	8.00	6.50	5.00
7.00	7.40	6.00	6.60
9.40	8.00	6.00	5.00
7.00	7.90	6.00	4.50
7.00	8.50	6.00	7.10
6.70	8.00	6.70	5.00
6.20	8.00	5.10	5.00
7.00	7.70	6.00	4.70

Table 5-56: Ultimate updated process times for the processes for machines 1,2,3 and 4 where all machines are subjected to deviations with respect to the discrete time-counter κ.

	$p_1(k) - p_1$	$p_2(k) - p_2$	$p_3(k) - p3$	$p_4(k) - p4$
$k = 1$	**2.30**	0	**0.50**	0
$k = 2$	0	**-0.60**	0	**1.60**
$k = 3$	**2.40**	0	**0**	0
$k = 4$	0	**-0.10**	0	**-0.50**
$k = 5$	0	**0.50**	0	**2.10**
$k = 6$	**-0.30**	0	**0.70**	0
$k = 7$	**-0.80**	0	**-0.90**	0
$k = 8$	0	**-0.30**	0	**-0.30**
$\sum$	+3.60	-0.50	+0.30	+2.90

Table 5-57: Differences between the updated ultimate process times with respect to the nominal process times. In bold the processes taken by the final scheduler.

drawn about the above made realizations. $\sum_{total} = +3.60 - 0.50 + 0.30 + 2.90 = +6.30$ and $\sum_{total} = +4.70 - 0.70 - 0.20 + 1.80 = +5.60$ for respectively the final schedule and the initial schedule. Subsequently it can told that $\sum_{dif} = +6.30 - 5.60 = +0.70$. The results for the realization is shown in 5-59, it can be observed it has a significant beneficial effect with almost a spare of 10 time-units.

Repeating this for 6 realizations gives the results shown in table 5-60. For the system with all machines deviating, a montecarlo simulation is done with $n = 100$. The comparison is done

	$p_1(k) - p_1$	$p_2(k) - p_2$	$p_3(k) - p3$	$p_4(k) - p4$
$k = 1$	**2.30**	0	0.50	**0**
$k = 2$	0	**-0.60**	**0**	1.60
$k = 3$	**2.40**	0	0	**0**
$k = 4$	0	**-0.10**	**0**	-0.50
$k = 5$	**0**	0.50	0	**2.10**
$k = 6$	-0.30	**0**	**0.70**	0
$k = 7$	-0.80	**0**	**-0.90**	0
$k = 8$	**0**	-0.30	0	**-0.30**
$\sum$	+4.70	-0.70	-0.20	+1.80

Table 5-58: Differences between the updated ultimate process times with respect to the nominal process times. In bold the processes taken by the initial scheduler.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule	220.60	+0.70	221.30
Updated schedule	210.70	+0.70	210.70

Table 5-59: Comparison of the shown realization of the sum of the endtimes between the initial schedule with fixed route and order and updated process times, and the final schedule with updated schedule.

	FVAL	$\sum_{dif}$	FVAL_{new}
Initial schedule(1)	216.60	+0.60	217.20
Updated schedule(1)	212.10	+0.60	212.70
Initial schedule(2)	203.70	+4.00	203.70
Updated schedule(2)	203.70	+4.00	203.70
Initial schedule(3)	193.40	-0.50	192.90
Updated schedule(3)	189.60	-0.50	189.60
Initial schedule(4)	204.40	-2.10	202.30
Updated schedule(4)	187.90	-2.10	187.90
Initial schedule(5)	207.30	+1.50	208.70
Updated schedule(5)	204.00	+1.50	204.00
Initial schedule(6)	196.30	+1.30	197.60
Updated schedule(6)	195.30	+1.30	195.30

Table 5-60: Comparison of 6 realizations of the sum of the endtimes between the schedule with fixed route and order but updated process times, and the final schedule with updated process times.

the same as in table 5-60 with the values for FVAL_{new}. One example for the first realization in table 5-60. The FVAL_{new} is 217.20 for the case where the initial schedule is obtained, and 212.70 where the schedule is adapted. Now the time gained is done by a percentage as:

$$\frac{212.20 - 217.20}{217.20} = -2.30\% \tag{5-8}$$

Such that it can be concluded the adaptive scheduler took care of a decrease of 2.30% in time-units. The montecarlo simulation had as a result the shown table in 5-61. The results are

$\text{FVAL}_{new} - FinalScheduler$	$\text{FVAL}_{new} - Initialscheduler$	$\sum_{dif}$	Decrease in cost(%)
196.67	202.02	+0.97	-2.11%

Table 5-61: Montecarlo simulation of the time-based updated with $n = 100$ with average value of the final scheduler its cost, average value of the initial scheduler its cost, average $\sum_{dif}$ and average decrease in total endtime with respect to the initial schedule in %.

nice, but somehow very marginal. There is an expected decrease in the total time of the sum of the endstimes of the jobs. However it is not very large. A possible explanation could be the system is quite simplistic. Not a lot of different routes or ordering can be adjusted where it becomes difficult for the system to adapt the schedule to certain disturbances. Another reason could be the system is too random, where the times deviate so randomly, the system can not make a logical update. In other words, the reasoning of adjusting a route is based on the expectation of the consecutive routes being the nominal process times. Though, if each of the routes deviate too much, it is not reasonable to change the route, because the next process could also be randomly shorter or longer. Still, the result is a lot better than the 1.21% for the montecarlo simulation of the cyclic updated system. This could be due to the fact in the cyclic updated behaviour, a whole cycle is fixed in which for instance both processes in machine 1 or 2 and machine 3 or 4, will be delayed or forwarded, where in the time-updated scheduler, one could still adapt to the situation when knowing machine 1 or 2 is taking longer.

Chapter 6

Conclusions and discussion

6-1 Conclusion

First of all the simulations, estimations and predictions are concluded. The simulations are made quite straigthforward and tuned with the parameters k_0 and b_0 to the assigned processing times. Research is done for the addition of process and measurements noise, as well as tuning the memory factor ζ. It can be concluded a too high covariances of the process nosie, will let the system deviate so much, the system its dynamics are almost neglible. Where a choice of the memory factor, will lead to almost no influence of the process noise when tuning it too close to 0, but will lead to almost no influence to the process noise when tuning it too close to 1. The trade-off is made between the choice of the value of the covariance of the process noise and the value of the memory factor, to obtain deviating processes which are still to a certain point predictable. Afterwards the estimations are made by the use of an extended Kalman filter, which as is shown in various figures and tables gives a very nice estimate of the states of the subsystems. Varying the measurement noise however, will lead to less accurate estimations, but nevertheless still sufficiently well estimations. However, the estimation of the last state $x_3(\kappa)$ becomes less and less accurate when increasing the measurement noise. An off-set can be observed and after a while the whole estimation is off. The off-set is mainly caused by the insufficiently well accurate measurements of the system, where the whole estimation being off, is caused by the term influenced by the noise becoming closer to 0, where as a result the noise can not be recognized anymore since the influence is nihil. The thereby followed predictions are made for the simulations and estimations with assigned covariance matrices and value for the memory factor. The predictions are concluded to be suitable, and convert to the real process time when κ increases. However, in some cases, the prediction give a less suitable prediction at the early stages of the predictions as can be concluded in the tables shown for 10 realizations per process for the prediction per discrete time κ.

The scheduler obtained gives by means of a switching max-plus approach optimal schedules. Initially, the optimal schedule obtained is the statical scheduler, which in comparison with randomly modes picking, gives a nice result as can be concluded by a montecarlo simulation.

Afterwards, the scheduler is updated with respect to a cyclic based update, and a time-based update. 4 approaches per update are discussed, namely approach 1 where a disruption takes place, approach 2 where machines 1 and 2 deviate in processing times, approach 3 where machines 3 and 4 deviate in processing times, and approach 3 where all machines deviate in processing times. A part from the results, quite nice insights in the systems its behaviour can be observed when looking at several GANTT-charts with respect to all the approaches.

Both the systems respond well to the disruption case, where the time-based schedule can change itself within the mode to machine 3 or 4 where the lowest processing time takes place, and where the cyclic based schedule is not able to do that because of the fact the modes are fixed per cycle. Nevertheless both schedulers in the disruptions cases make a very nice improvement of the cost function. Important as well, is the note of the disadvantage of the cyclic updated behaviour. Namely, it is shown how the scheduler adapts itself with changing modes in the past, which in reality is not possible. As a conclusion a time-based approach for the event driven scheduler is preferred.

For the second, third and fourth approach, conclusions can be made in all approaches the updated schedule takes care of an improvement of the cost in a lot of cases, however these improvements are small. In a minor amount of the cases the updated schedule is concluded to have a worsening of the cost. A possible explanation is given as the modes switch to certain other modes for an improvement of the costs, but when realizing the processes in these other modes, the realizations turn out to take much longer than the expected nominal processing times such that the mode change was in hindsight a bad choice.

Finally, a montecarlo simulation is done for both the cylic- and time-based updates where all machines are subjected to deviating processing times. As can be concluded by the results shown in the tables for the realizations of the updated schedulers versus the initial schedulers, the updated schedulers will gain some negative cost when updating the schedule. The results for the montecarlo simulation when looking at the cyclic-updated behaviour and time-updated behaviour both do not have a significant difference. For the cyclic updated scheduler, the average decrease was -1.21% for the cost and for the time based updated the average decrease was -2.11%. It still is a decrease, such that compared with the overal cost it gives some negative cost. However, a possible explanation could be given the decrease is not major due to the fact the system does not have a lot of rerouting and reordering decision variables to its property, since there are only 4 routes to take and ordering takes place only in the last machine.

6-2 Discussion and further research

6-2-1 Discussion

The discussion is mainly focussed on the weak spots of this thesis research. All the discussion points are separated in different paragraphs to remain having oversight of the points. Further research thereafter, will be more focussed on the extensions of this research, instead of the weak spots of this research. Of course, the weak spots could lead to further research.

To begin, this thesis is essentially focussed on the idea of updating with respect to the iterative communication of information between a scheduler and its subsystems. The results and conclusions are provided by the means of a simple toy example of a beer brewery. However, the case study provided, is quite simplistic. Only 4 routes can be chosen accordingly, where as an addition of the simplicity, only ordering can occur in the last machine. This will lead to a system which can only vary on a minimal amount of routes and ordering variables, which as a result becomes stuck in its own system quite fast. To elaborate that last sentence a little bit more, if a certain job can only adjust its route on 3 other routes, and the ordering only in the last machine, it is not most likely the results of improvements will be very major. The simplicity of the case study provided makes therefore room for the question if any right conclusion with respect to the extent of improvements can be drawn from such a simplistic system. If the system would have for instance 3 mashing tanks, 3 brewing tanks and 2 fermentation tanks, or ordering in not only the last machine, but also in the machines where the brewing occurs would have been, the overal updated scheduler could have a much greater amount of choices, and therefore maybe a lot more improvement than shown in this case study.

Furthermore, the thesis itself is basically a simulation. Realizations are obtained of certain subsystems, which are afterwards encountered in an updated schedule. They are encountered in the updated schedule by means of the predictions. The behaviour of these predictions is shortly discussed as they convert to the real process time when time increases. However the reliability in this thesis is not derived in statistical properties, such that it can be concluded for instance a certain prediction can be considered fixed within a certain time-span when being for example 3 time-units before the predicted endtime. Not only the reliability of the predictions must be encountered into certain numerical statements, with respect to for example statistical properties, but also need to be compared with real systems. If the system does get nice results with the prediction of the simulation, this does not necessarily conclude directly the system does get nice results when applying the same methods to real systems. In other words, the prediction of real systems could be much less reliable, and the question could be asked if the conclusions drawn in this thesis are robust for systems where the predictions would be less reliable.

To continue, conclusions are drawn about the optimality of an iterative method which updates every time-step κ in comparison with an initial optimal schedule. However, one of the problems discussed in the introduction, is that an continuously updated model can be computationally hard. In this thesis research, the iterative method is not compared with a continuously updating model. Nothing can be concluded about the optimality of the iterative scheduler with respect to a continuously updating scheduler, while this is an important point of interest. The beneficial effect of the iterative scheduler would pale in comparison, if it turns out continuously updating the schedule would improve the system its cost function with a scale of 10 for instance. To extend this paragraph about comparisons, one could also ask questions about the manner which is used to compare the initial schedule with the final schedule. Namely the differences in nominal processing times are summed up, and afterwards the difference between these sums of respectively the initial schedule and the final schedule are calculated to state the time already gained or lost with shorter or longer processing times only by the processes rather than the scheduling. Another approach could be to scale the processes which the initial schedule does take as nominal processing time with the same value as the scale the value of the processing time in the updated scheduler is increased or de-

screased. A third approach could be to seed the noise sequences the same for machines of the same processes, such that if the initial schedule does not take the same machines as the final schedule per cycle k, it still undergoes the same noise sequence.

Ultimately, a lot of assumptions and choices have been made for simplicity, such as neglecting the transportation times between the processes, or the simplications made regarding the system dynamics of the processes. Of course, this is more a toy example and very specific and precise model properties are not necessary. However, if one want to obtain very accurate results of the beneficial effect of the updated scheduler, these things could be improved. A part from implementing transportation times, elaborate subsystems a lot more precisely, also the noise sequences chosen for the simulation and estimation are quite simplistic. Zero-mean white noise signals could be replaced by noise sequences who are not very 'clean'. Also the noise is assumed to be additive, but integrated noise would also be worthy to look at for the results and conclusions to become a little bit wider and more robust.

6-2-2 Further research

In the discussion section, already a lot of points are made where it opens way for further research. In this section some points of the discussion section are elaborated for further research, as well as new problems will be given for interest in further research. The same as done in the previous section, the further research point will be discussed per paragraph.

As is already mentioned in the above discussion, the iteratively updated scheduler is only compared with the initial optimal scheduler and not with a continuously updated scheduler. This can be done in future research by making the discrete time-step κ smaller and smaller, and observe the beneficial effect of the updated scheduler versus the computational power and time it needs for the solution. Essentially a continuously updating scheduler is the same as making the discrete time-counter κ infinitely small, such that it updates every time-counter κ, which when κ is almost equal to zero, can be seen as continuously updating the scheduler.

As an extension on the previous paragraph, prediction can also be discussed within this discrete time-counter κ. Further research could be scoped upon the prediction of these subsystems when not using additive zero-mean white noise signals, but for instance integrated coloured noise. It will rise the question what the change of noise will do to the extended Kalman filter and therefore also to the predictions. Moreover, as can already be observed in the tables with results of the predictions, the predictions are often not very accurate when being in an early stage of the prediction. In further research, an interest could be to put a weight on the prediction when increasing in time such that the schedule does not switch too fast when having a very unreliable prediction of an early stage process. To extend in the field of predictions, one could ask themselves if prediction of delays or forwards can not be reframed into nominal processing times with the use of control. This is a very worthy to look at point of interest. If a process is delayed, control can be used to let the system still float to the nominal process time, such that rescheduling is unnecessary. If the control law can not be fulfilled anymore, rescheduling can be done with respect to minimizing the costs. For instance, you would rather like having a train driving faster to arrive on time, than the train remaining its path but rescheduling the whole timetable due to this train.

Also mentioned in the discussion paragraph is the simplicity of the provided case study. Further research could want to extend the methods discussed in this thesis to systems which are much more complex, and observe if the same improvements can be made with respect to the cost, or even better improvements. One could imagine a system as a railway network with 100 different train has much more other route- and order-decision which can be made than a 4 route and 1 order beer brewing case study.

Appendix A

Routing and ordering constraints

A-1 Constraint matrices routing

The first constraint:

$$\begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \end{bmatrix} \geq v_1(k) \otimes \begin{bmatrix} \varepsilon \\ \varepsilon \\ p_1 + z_1(k) \\ \varepsilon \\ p_3 + z_3(k) \end{bmatrix} = \begin{bmatrix} \beta \\ \beta \\ p_1 + z_1(k) + \beta - \beta v_1(k) \\ \beta \\ p_3 + z_3(k) + \beta - \beta v_1(k) \end{bmatrix} \tag{A-1}$$

Rearranging the equation:

$$\begin{bmatrix} -z_1(k) \\ -z_2(k) \\ z_1(k) - z_3(k) - \beta v_1(k) \\ -z_4(k) \\ z_3(k) - z_5(k) - \beta v_1(k) \end{bmatrix} \leq \begin{bmatrix} -\beta \\ -\beta \\ -p_1 - \beta \\ -\beta \\ -p_3 - \beta \end{bmatrix} \tag{A-2}$$

This is equal to:

$$\underbrace{\begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}}_{A_0(l=1)} z(k) + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\beta & 0 & 0 & 0 \end{bmatrix}}_{V_0(l=1)} v(k) \leq \underbrace{\begin{bmatrix} -\beta \\ -\beta \\ -p_1 - \beta \\ -\beta \\ -p_3 - \beta \end{bmatrix}}_{b_0(l=1)} \tag{A-3}$$

Continuing making this for each mode gives:

$$\underbrace{\begin{bmatrix}
0 & A_0(l=1) & 0 & \dots & 0 & 0 & V_0(l=1) & 0 & \dots & 0 \\
0 & \vdots & 0 & \dots & 0 & 0 & \vdots & 0 & \dots & 0 \\
0 & A_0(l=4) & 0 & \dots & 0 & 0 & V_0(l=4) & 0 & \dots & 0 \\
\hline
0 & 0 & A_0(l=1) & 0 & \dots & 0 & 0 & V_0(l=1) & \dots & 0 \\
0 & 0 & \vdots & 0 & \dots & 0 & 0 & \vdots & \dots & 0 \\
0 & 0 & A_0(l=4) & 0 & \dots & 0 & 0 & V_0(l=4) & \dots & 0 \\
\hline
\vdots & & & \ddots & & & & \ddots & & \vdots \\
\hline
0 & \dots & 0 & 0 & A_0(l=1) & 0 & \dots & 0 & V_0(l=1) & 0 \\
0 & \dots & 0 & 0 & \vdots & 0 & \dots & 0 & \vdots & 0 \\
0 & \dots & 0 & 0 & A_0(l=4) & 0 & \dots & 0 & V_0(l=4) & 0
\end{bmatrix}}_{F_1} q \leq \underbrace{\begin{bmatrix}
b_0(l=1) \\ \vdots \\ b_0(l=4) \\ \hline b_0(l=1) \\ \vdots \\ b_0(l=4) \\ \hline \vdots \\ \hline b_0(l=1) \\ \vdots \\ b_0(l=4)
\end{bmatrix}}_{p_1} \tag{A-4}$$

With all the zeros in the matrix of appropriate size. Having the same procedure with the next constraints involving $z(k-1)$ and $u(k)$, it becomes:

$$\underbrace{\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}}_{A_1(l=1)} z(k-1) + \underbrace{\begin{bmatrix}
-1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & -1
\end{bmatrix}}_{A_{1,0}(l=1)} z(k) + \underbrace{\begin{bmatrix}
-\beta & 0 & 0 & 0 \\
-\beta & 0 & 0 & 0 \\
-\beta & 0 & 0 & 0 \\
-\beta & 0 & 0 & 0 \\
-\beta & 0 & 0 & 0
\end{bmatrix}}_{V_1(l=1)} v(k-1) \leq \underbrace{\begin{bmatrix}
-p_1 - \beta \\ -\beta \\ -p_3 - \beta \\ \beta \\ -p_5 - \beta
\end{bmatrix}}_{b_1(l=1)} \tag{A-5}$$

Put the above matrices in appropriate places with respect to the vector z until event-step $k + N_p$ and one will get a similar matrix as F_1 such that a big matrix can be obtained, also taking the different modes into account:

$$F_2 q \leq p_2 \tag{A-6}$$

And finally the same procedure for the constraint with respect to the input:

$$\begin{bmatrix}
-1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & -1
\end{bmatrix} z(k) + \begin{bmatrix}
-\beta \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix} v(k) + \begin{bmatrix}
1 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix} u(k) \leq \begin{bmatrix}
-\beta \\ -\beta \\ -\beta \\ -\beta \\ -\beta
\end{bmatrix} \tag{A-7}$$

A-2 Constraint matrices ordering

Ordering 1:

$$\begin{bmatrix} -z_5(k)+z_5(k+1)-\beta s_{0,1}(k) \\ -z_5(k)+z_5(k+2)-\beta s_{0,2}(k) \\ -z_5(k)+z_5(k+3)-\beta s_{0,3}(k) \\ -z_5(k)+z_5(k+4)-\beta s_{0,4}(k) \\ -z_5(k)+z_5(k+5)-\beta s_{0,5}(k) \\ -z_5(k)+z_5(k+6)-\beta s_{0,6}(k) \\ -z_5(k)+z_5(k+7)-\beta s_{0,7}(k) \end{bmatrix} \leq \begin{bmatrix} -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \end{bmatrix} \tag{A-8}$$

Ordering 2:

$$\begin{bmatrix} -z_5(k+1)+z_5(k)-\beta s_{0,1}(k) \\ \hline -z_5(k+1)+z_5(k+2)-\beta s_{1,2}(k+1) \\ -z_5(k+1)+z_5(k+2)-\beta s_{1,3}(k+1) \\ -z_5(k+1)+z_5(k+2)-\beta s_{1,4}(k+1) \\ -z_5(k+1)+z_5(k+2)-\beta s_{1,5}(k+1) \\ -z_5(k+1)+z_5(k+2)-\beta s_{1,6}(k+1) \\ -z_5(k+1)+z_5(k+2)-\beta s_{1,7}(k+1) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ \hline -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \end{bmatrix} \tag{A-9}$$

Ordering 3:

$$\begin{bmatrix} -z_5(k+2)+z_5(k)-\beta s_{0,2}(k) \\ -z_5(k+2)+z_5(k+1)-\beta s_{1,2}(k+1) \\ \hline -z_5(k+2)+z_5(k+3)-\beta s_{2,3}(k+2) \\ -z_5(k+2)+z_5(k+4)-\beta s_{2,4}(k+2) \\ -z_5(k+2)+z_5(k+5)-\beta s_{2,5}(k+2) \\ -z_5(k+2)+z_5(k+6)-\beta s_{2,6}(k+2) \\ -z_5(k+2)+z_5(k+7)-\beta s_{2,7}(k+2) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ -p_5 \\ \hline -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \end{bmatrix} \tag{A-10}$$

Ordering 4:

$$\begin{bmatrix} -z_5(k+3)+z_5(k)-\beta s_{0,3}(k) \\ -z_5(k+3)+z_5(k+1)-\beta s_{1,3}(k+1) \\ -z_5(k+3)+z_5(k+2)-\beta s_{2,3}(k+2) \\ \hline -z_5(k+3)+z_5(k+4)-\beta s_{3,4}(k+3) \\ -z_5(k+3)+z_5(k+5)-\beta s_{3,5}(k+3) \\ -z_5(k+3)+z_5(k+6)-\beta s_{3,6}(k+3) \\ -z_5(k+3)+z_5(k+7)-\beta s_{3,7}(k+3) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ -p_5 \\ -p_5 \\ \hline -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \end{bmatrix} \tag{A-11}$$

Ordering 5:

$$\begin{bmatrix} -z_5(k+4)+z_5(k)-\beta s_{0,4}(k) \\ -z_5(k+4)+z_5(k+1)-\beta s_{1,4}(k+1) \\ -z_5(k+4)+z_5(k+2)-\beta s_{2,4}(k+2) \\ -z_5(k+4)+z_5(k+3)-\beta s_{3,4}(k+3) \\ \hline -z_5(k+4)+z_5(k+5)-\beta s_{4,5}(k+4) \\ -z_5(k+4)+z_5(k+6)-\beta s_{4,6}(k+4) \\ -z_5(k+4)+x_5(k+7)-\beta s_{4,7}(k+4) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ \hline -\beta-p_5 \\ -\beta-p_5 \\ -\beta-p_5 \end{bmatrix} \tag{A-12}$$

Ordering 6:

$$\begin{bmatrix} -z_5(k+5)+z_5(k)-\beta s_{0,5}(k) \\ -z_5(k+5)+z_5(k+1)-\beta s_{1,5}(k+1) \\ -z_5(k+5)+z_5(k+2)-\beta s_{2,5}(k+2) \\ -z_5(k+5)+z_5(k+3)-\beta s_{3,5}(k+3) \\ -z_5(k+5)+z_5(k+4)-\beta s_{4,5}(k+3) \\ \hline -z_5(k+5)+z_5(k+6)-\beta s_{5,6}(k+5) \\ -z_5(k+5)+z_5(k+7)-\beta s_{5,7}(k+5) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ \hline -\beta-p_5 \\ -\beta-p_5 \end{bmatrix} \tag{A-13}$$

Ordering 7:

$$\begin{bmatrix} -z_5(k+6)+z_5(k)-\beta s_{0,6}(k) \\ -z_5(k+6)+z_5(k+1)-\beta s_{1,6}(k+1) \\ -z_5(k+6)+z_5(k+2)-\beta s_{2,6}(k+2) \\ -z_5(k+6)+z_5(k+3)-\beta s_{3,6}(k+3) \\ -z_5(k+6)+z_5(k+4)-\beta s_{4,6}(k+4) \\ -z_5(k+6)+z_5(k+5)-\beta s_{5,6}(k+5) \\ \hline -z_5(k+6)+z_5(k+7)-\beta s_{6,7}(k+6) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ \hline -\beta-p_5 \end{bmatrix} \tag{A-14}$$

Ordering 8:

$$\begin{bmatrix} -z_5(k+7)+z_5(k)-\beta s_{0,7}(k) \\ -z_5(k+7)+z_5(k+1)-\beta s_{1,7}(k+1) \\ -z_5(k+7)+z_5(k+2)-\beta s_{2,7}(k+2) \\ -z_5(k+7)+z_5(k+3)-\beta s_{3,7}(k+3) \\ -z_5(k+7)+z_5(k+4)-\beta s_{4,7}(k+4) \\ -z_5(k+7)+z_5(k+5)-\beta s_{5,7}(k+5) \\ -z_5(k+7)+z_5(k+6)-\beta s_{6,7}(k+6) \end{bmatrix} \leq \begin{bmatrix} -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \\ -p_5 \end{bmatrix} \tag{A-15}$$

Appendix B

MATLAB-files

B-1 Scheduler MATLAB-files

B-1-1 Scheduler main file

```
%% Begin file
for z = 1:100


clearvars -except Montecarlo z
clc
addpath Mashing
addpath Brewing
%% Parameters and matrices
beta = -30000;
Np = 7;
length  = 5*(Np+2)+4*(Np+2)+(Np+1)+sum(1:Np);
Zvec = zeros(1,4*(Np+1));

p1 = ones(Np+1,1)*7; % Processing times per machine
p2 = ones(Np+1,1)*8;
p3 = ones(Np+1,1)*6;
p4 = ones(Np+1,1)*5;
p5 = ones(Np+1,1)*1;

seed1  = randi(10000,Np+1,1); % Seeds for the random processes
seed2  = randi(10000,Np+1,1);
seed3  = randi(10000,Np+1,1);
seed4  = randi(10000,Np+1,1);

```

```
26 %% Initializing for t=0, the statical scheduler
27 Ysimend = 0;
28 Ystart  = 0;
29 t = 0;
30
31 [Y, modes, Ysimend, Ystart, FVAL, p1, p2, p3, p4, p5, Z, X] =
      dynScheduleTimedriven(p1, p2, p3, p4, p5, [], [], [], [], [],
      [], [], [], Ysimend, Ystart, Np, length, t, modes, beta, Zvec,
      seed1, seed2, seed3, seed4);
32 % Function file for the initial schedule
33
34 Veq5 = zeros((Np+1)*2,length); % Prebuild matrices for the mode
      constraints when time counter is passed starting time
35 beq5 = zeros((Np+1)*2,1);
36
37 Veq6 = zeros((Np+1)*2,length); % Prebuild matrices for the mode
      constraints when time counter is passed starting time
38 beq6 = zeros((Np+1)*2,1);
39
40 Veq7 = zeros((Np+1)*2,length); % Prebuild matrices for the mode
      constraints when time counter is passed starting time
41 beq7 = zeros((Np+1)*2,1);
42
43 Veq8 = zeros((Np+1)*2,length); % Prebuild matrices for the mode
      constraints when time counter is passed starting time
44 beq8 = zeros((Np+1)*2,1);
45
46 Modes = modes; % Initial modes
47 Zvec = Z(:)'; % Vector for previous modes, such that innerproduct
      brings negative cost and modes stay the same
48 Xtotal = X;    % Value of the optimization vector
49 Ystarttotal = Ystart; % Starting times of the jobs
50 Ysimendtotal = Ysimend; % End times of the jobs
51
52 p1check = p1;
53 p2check = p2;
54 p3check = p3;
55 p4check = p4;
56 p5check = p5;
57
58 %% Scheduler update with respect to ordering and routing
59
60 for t = 1:50
61
62 [Y, modes, Ysimend, Ystart, FVAL, p1, p2, p3, p4, p5, Z, X] =
      dynScheduleTimedriven(p1, p2, p3, p4, p5, Veq5, beq5, Veq6, beq6
```

```
    , Veq7, beq7, Veq8, beq8, Ysimend, Ystart, Np, length, t, modes
    , beta, Zvec, seed1, seed2, seed3, seed4);
% Optimization algorithm per discrete time counter t

p1check(1:(Np+1), t+1) = p1; % Updated process times M1 per
    discrete time counter t
p2check(1:(Np+1), t+1) = p2; % Updated process times M2 per
    discrete time counter t
p3check(1:(Np+1), t+1) = p3; % Updated process times M3 per
    discrete time counter t
p4check(1:(Np+1), t+1) = p4; % Updated process times M4 per
    discrete time counter t
p5check(1:(Np+1), t+1) = p5; % Updated process times M5 per
    discrete time counter t(useless)


Ystarttotal((1+5*t):(5*t+5),1:Np+1) = Ystart; % Total matrix for
    starting times per t
Ysimendtotal((1+5*t):(5*t+5), 1:Np+1) = Ysimend; % Total matrix for
     end times per t
Xtotal(1:length, t+1) = X; % Total matrix for optimization vector X
     per t
Modes(t+1,:) = modes; % Total matrix for modes vector per t
FcheckVAL(t) = FVAL; % Value of the cost function per t

Zvec = Z(:)'; % Previous modes for inner product of the modes
end
%% Simulation of the initial schedule and order with updated
    process times for comparison
% for t = 1:50
X1 = Xtotal(:,1);
modesX1 = Modes(1,:);
cycle = 0;
Zvec = Zvec';

[YstartX1, YsimendX1, modesX1, ZX1, checkFVAL, p1, p2, p3, p4, p5]
   = dynScheduleSim(p1, p2, p3, p4, p5, Np, length, cycle, modesX1,
     beta, Zvec,X1);

%
p1abs = p1(:,1)-7; % Difference vector p1 with nominal process
    times p1
p2abs = p2(:,1)-8; % Difference vector p2 with nominal process
    times p2
p3abs = p3(:,1)-6; % Difference vector p3 with nominal process
    times p3
```

```
93  p4abs = p4(:,1)-5; % Difference vector p4 with nominal process
        times p4
94
95  pdif = sum(p1abs)+sum(p2abs)+sum(p3abs)+sum(p4abs)  % Sum of the
        differences
96
97  P = [p1abs p2abs p3abs p4abs];
98
99  for k=1:8
100     modes = Modes(1,:);
101
102     if modes(k) == 1
103     p2abs(k) = 0;
104     p4abs(k) = 0;
105
106     elseif modes(k) ==2
107         p2abs(k) = 0;
108         p3abs(k) = 0;
109
110     elseif modes(k) == 3
111     p1abs(k) = 0;
112     p4abs(k) = 0;
113
114     elseif modes(k) == 4
115     p1abs(k) = 0;
116     p3abs(k) = 0;
117     end
118 end
119 pdifnew = sum(p1abs)+sum(p2abs) +sum(p3abs)+sum(p4abs)
120 sigma = pdif - pdifnew
121 FVAL
122 checkFVAL
123 % %% GANTT chart
124 % GANTT(YstartX1, YsimendX1, modesX1, Np, p1, p2, p3, p4, p5)
125
126 Montecarlo(z, 1:5) = [FVAL sigma pdif pdifnew checkFVAL];
127 end
```

B-1-2 Scheduler function file

```
1  function[Y, modes, Ysimend, Ystart, FVAL, p1, p2, p3, p4, p5, Z, X]
       = dynScheduleTimedriven(p1, p2, p3, p4, p5, Veq5, beq5, Veq6,
      beq6, Veq7, beq7, Veq8, beq8, Ysimend, Ystart, Np, length, t,
      modes, beta, Zvec, seed1, seed2, seed3, seed4)
2  p1real = p1; % Disruption case
3  p2real = p2; % Disruption case
4  p3real = p3; % Disruption case
```

```
5  p4real = p4; % Disruption case
6
7
8  %% Updating the processing times
9  if t > 0
10
11 for k = 1:Np+1
12     if modes(k) < 3 && t>Ystart(1,k)
13     Veq5((2*k-1):(2*k), (48+4*k):(49+4*k)) = eye(2); % Eliminate
           modes which can not be chosen anymore due to processes which
            have been started
14     beq5((2*k-1):(2*k), 1) = zeros(2,1);
15     p1(k,1) = p1real(k);%mashing1(ceil(t-Ystart(1,k)), seed1(k)); %
           Update process times machine 1
16
17     elseif  modes(k) > 2 && t>Ystart(2,k)
18             Veq6((2*k-1):(2*k), (46+4*k):(47+4*k)) = eye(2); %
                Eliminate modes which can not be chosen anymore due
                to processes which have been started
19             beq6((2*k-1):(2*k), 1) = zeros(2,1);
20             p2(k,1) = p2real(k);%mashing2(ceil(t-Ystart(2,k)),
                seed2(k));
21     end
22
23
24     if ((modes(k) == 1 || modes(k)==3) && t>Ystart(3,k))
25     Veq7(2*k-1,47+4*k) = 1;  % Eliminate modes which can not be
           chosen anymore due to processes which have been started
26     Veq7(2*k, 49+4*k)= 1;
27     beq7((2*k-1):(2*k), 1) = zeros(2,1);
28             p3(k,1) = brewing1(ceil(t-Ystart(3,k)), seed3(k));
29
30     elseif ((modes(k) == 2 || modes(k)==4) && t>Ystart(4,k))
31         Veq8(2*k-1,46+4*k) = 1; % Eliminate modes which can not be
               chosen anymore due to processes which have been started
32         Veq8(2*k, 48+4*k)= 1;
33         beq8((2*k-1):(2*k), 1) = zeros(2,1);
34         p4(k,1) = brewing2(ceil(t-Ystart(4,k)), seed4(k));
35     end
36 end
37
38 end
39
40
41 %% Matrices A0 for mode l=1 cycle k
42 A0_L1 = [0 0 0 0 0;0 0 0 0 0; 1 0 0 0 0;0 0 0 0 0; 0 0 1 0 0];
43 A0_L1 = A0_L1-eye(5);
```

```
44 V0_L1 = zeros(5,4);
45 V0_L1(:,1) = [0 0 -beta 0 -beta]';
46
47 F0_L1 = zeros(((Np+1)*5) , length);
48
49 for k = 1:Np+1
50
51 F0_L1((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L1;
52 F0_L1((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
       V0_L1;
53
54 b0_L1((1+5*k-5):(5*k),:) = [-beta -beta -beta-p1(k,:) -beta -beta-
       p3(k,:)]';
55 end
56 %% Matrices A0 for mode l=2 cycle k
57 A0_L2 = [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];
58 A0_L2 = A0_L2-eye(5);
59 V0_L2 = zeros(5,4);
60 V0_L2(:,2) = [0 0 0 -beta -beta]';
61
62 F0_L2 = zeros(((Np+1)*5) , length);
63
64 for k = 1:Np+1
65
66 F0_L2((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L2;
67 F0_L2((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
       V0_L2;
68
69 b0_L2((1+5*k-5):(5*k),:) = [-beta -beta -beta -beta-p1(k,:) -beta-
       p4(k,:)]';
70 end
71 %% Matrices A0 for mode l=3 cycle k
72 A0_L3 = [0 0 0 0 0;0 0 0 0 0;0 1 0 0 0;0 0 0 0 0; 0 0 1 0 0];
73 A0_L3 = A0_L3-eye(5);
74 V0_L3 = zeros(5,4);
75 V0_L3(:,3) = [0 0 -beta 0 -beta]';
76
77 F0_L3 = zeros(((Np+1)*5) , length);
78
79 for k = 1:Np+1
80
81 F0_L3((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L3;
82 F0_L3((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
       V0_L3;
83
84 b0_L3((1+5*k-5):(5*k),:) = [-beta -beta -beta-p2(k,:) -beta -beta-
       p3(k,:)]';
```

```
85  end
86  %% Matrices A0 for mode l=4 cycle k
87  A0_L4 = [0 0 0 0 0;0 0 0 0 0; 0 0 0 0 0;0 1 0 0 0; 0 0 0 1 0];
88  A0_L4 = A0_L4-eye(5);
89  V0_L4 = zeros(5,4);
90  V0_L4(:,4) = [0 0 0 -beta -beta]';
91
92  F0_L4 = zeros(((Np+1)*5) , length);
93
94  for k = 1:Np+1
95
96  F0_L4((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L4;
97  F0_L4((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
        V0_L4;
98
99  b0_L4((1+5*k-5):(5*k),:) = [-beta -beta -beta -beta-p2(k,:) -beta-
        p4(k,:)]';
100 end
101
102 %% Total constraint matrices for all the modes cycle k
103 F0 = [F0_L1;F0_L2;F0_L3;F0_L4];
104
105 b0_0 = [b0_L1; b0_L2; b0_L3; b0_L4];
106
107
108 %% Matrices A1 for mode l=1 cycle k
109 A1_L1 = [eye(5) -eye(5)];
110 A1_L1(5,:) = 0;
111 V1_L1 = zeros(5,4);
112 V1_L1(:,1) = -ones(5,1)*beta;
113
114 F1_L1 = zeros((Np+1)*5, length);
115 b1_L1 = zeros((Np+1)*5, 1);
116 for k = 1:Np
117
118 F1_L1((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L1;
119 F1_L1((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
        V1_L1;
120
121 b1_L1((1+5*k-5):(5*k), :) = [-beta-p1(k,:) -beta -beta-p3(k,:) -
        beta -beta]';
122 end
123
124 %% Matrices for A1 for mode l=2 cycle k
125
126 A1_L2 = [eye(5) -eye(5)];
127 A1_L2(5,:) = 0;
```

```
128 V1_L2 = zeros(5,4);
129 V1_L2(:,2) = -ones(5,1)*beta;
130
131 F1_L2 = zeros((Np+1)*5, length);
132 b1_L2 = zeros((Np+1)*5, 1);
133 for k = 1:Np
134
135 F1_L2((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L2;
136 F1_L2((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
       V1_L2;
137
138 b1_L2((1+5*k-5):(5*k), :) = [-beta-p1(k,:) -beta -beta -beta-p4(k
       ,:) -beta]';
139 end
140
141 %% Matrices for A1 for mode l=3 cycle k
142
143 A1_L3 = [eye(5) -eye(5)];
144 A1_L3(5,:) = 0;
145 V1_L3 = zeros(5,4);
146 V1_L3(:,3) = -ones(5,1)*beta;
147
148 F1_L3 = zeros((Np+1)*5, length);
149 b1_L3 = zeros((Np+1)*5, 1);
150 for k = 1:Np
151
152 F1_L3((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L3;
153 F1_L3((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
       V1_L3;
154
155 b1_L3((1+5*k-5):5*k, :) = [-beta -beta-p2(k,:) -beta-p3(k,:) -beta
       -beta]';
156 end
157
158 %% Matrices for A1 for mode l=4 cycle k
159
160 A1_L4 = [eye(5) -eye(5)];
161 A1_L4(5,:) = 0;
162 V1_L4 = zeros(5,4);
163 V1_L4(:,4) = -ones(5,1)*beta;
164
165 F1_L4 = zeros((Np+1)*5, length);
166 b1_L4 = zeros((Np+1)*5, 1);
167 for k = 1:Np
168
169 F1_L4((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L4;
```

```
170 F1_L4((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
        V1_L4;
171
172 b1_L4((1+5*k-5):5*k, :) = [-beta -beta-p2(k,:) -beta -beta-p4(k,:)
        -beta]';
173 end
174
175 %% Total constraint matrices for all the modes cycle k-1
176 F1 = [F1_L1;F1_L2;F1_L3;F1_L4];
177
178 b1_1 = [b1_L1; b1_L2; b1_L3; b1_L4];
179
180 %% Constraints for the input mode l=1
181 AU_L1 = -eye(5);
182 VU_L1 = zeros(5,4);
183 VU_L1(:,1) = [-beta 0 0 0 0]';
184 U_L1 = [1 0 0 0 0]';
185
186 FU_L1 = zeros((Np+1)*5, length);
187
188 for k = 1:(Np+1)
189 FU_L1((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L1;
190 FU_L1((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L1;
191 FU_L1((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L1;
192
193 bU_L1((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
194 end
195
196 %% Constraint for the input mode l=2
197 AU_L2 = -eye(5);
198 VU_L2 = zeros(5,4);
199 VU_L2(:,2) = [-beta 0 0 0 0]';
200 U_L2 = [1 0 0 0 0]';
201
202 FU_L2 = zeros((Np+1)*5, length);
203
204 for k = 1:(Np+1)
205 FU_L2((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L2;
206 FU_L2((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L2;
207 FU_L2((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L2;
208
209 bU_L2((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
210 end
211
212 %% Constraints for the input mode l=3
213 AU_L3 = -eye(5);
214 VU_L3 = zeros(5,4);
```

```
215 VU_L3(:,3) = [0 -beta 0 0 0]';
216 U_L3 = [0 1 0 0 0]';
217
218 FU_L3 = zeros((Np+1)*5, length);
219
220 for k = 1:(Np+1)
221 FU_L3((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L3;
222 FU_L3((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L3;
223 FU_L3((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L3;
224
225 bU_L3((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
226 end
227
228
229 %% Constraints for the input mode l=4
230
231 AU_L4 = -eye(5);
232 VU_L4 = zeros(5,4);
233 VU_L4(:,4) = [0 -beta 0 0 0]';
234 U_L4 = [0 1 0 0 0]';
235
236 FU_L4 = zeros((Np+1)*5, length);
237
238 for k = 1:(Np+1)
239 FU_L4((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L4;
240 FU_L4((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L4;
241 FU_L4((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L4;
242
243 bU_L4((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
244 end
245
246 %% Total constraint matrices for all input modes at cycle k
247 FU = [FU_L1; FU_L2; FU_L3; FU_L4];
248
249 bU = [bU_L1; bU_L2; bU_L3; bU_L4];
250
251 %% Orderings constraint for the process of machine 5 normal binary
    var
252 Aordk  = zeros(Np, length);
253 Aordk1 = zeros(Np-1, length);
254 Aordk2 = zeros(Np-2, length);
255 Aordk3 = zeros(Np-3, length);
256 Aordk4 = zeros(Np-4, length);
257 Aordk5 = zeros(Np-5, length);
258 Aordk6 = zeros(Np-6, length);
259
```

```
260 for k = 1:Np                                % Matrices for the non-
       conjunct cases of the binary variables
261 Aordk(k, 10+5*k) = 1;
262 Aordk(:,10) = -1;
263 Aordk(k, length-sum(1:Np)+k) = -beta;
264 bordk(k,1) = -beta-p5(k);
265 end
266
267 for k = 2:Np
268 Aordk1(k-1,10+5*k) = 1;
269 Aordk1(:,15) = -1;
270 Aordk1(k-1, length-sum(1:Np)+Np+(k-1)) = -beta;
271 bordk1(k-1,1) =-beta-p5(k);
272 end
273
274 for k = 3:Np
275 Aordk2(k-2,10+5*k) = 1;
276 Aordk2(:,20) = -1;
277 Aordk2(k-2, length-sum(1:Np)+Np+(Np-1)+(k-2)) = -beta;
278 bordk2(k-2,1) =-beta-p5(k);
279 end
280
281 for k = 4:Np
282 Aordk3(k-3,10+5*k) = 1;
283 Aordk3(:,25) = -1;
284 Aordk3(k-3, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(k-3)) = -beta;
285 bordk3(k-3,1) =-beta-p5(k);
286 end
287
288 for k = 5:Np
289 Aordk4(k-4,10+5*k) = 1;
290 Aordk4(:,30) = -1;
291 Aordk4(k-4, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(Np-3)+(k-4)) = -beta
       ;
292 bordk4(k-4,1) =-beta-p5(k);
293 end
294
295 for k = 6:Np
296 Aordk5(k-5,10+5*k) = 1;
297 Aordk5(:,35) = -1;
298 Aordk5(k-5, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(k-5))
       = -beta;
299 bordk5(k-5,1) =-beta-p5(k);
300 end
301
302 for k = 7:Np
303 Aordk6(k-6,10+5*k) = 1;
```

```
304 Aordk6(:,40) = -1;
305 Aordk6(k-6, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)
        +(k-6)) = -beta;
306 bordk6(k-6,1) =-beta-p5(k);
307 end
308
309 Fordtot = [Aordk; Aordk1; Aordk2; Aordk3; Aordk4; Aordk5; Aordk6];
310 bordtot = [bordk; bordk1; bordk2; bordk3; bordk4; bordk5; bordk6];
311
312 %% Orderings constraint for the process of machine 5 normal binary
        var
313 Aordinv     = zeros(1,length);
314 Aordinv1    = zeros(2,length);
315 Aordinv2    = zeros(3,length);
316 Aordinv3    = zeros(4,length);
317 Aordinv4    = zeros(5,length);
318 Aordinv5    = zeros(6,length);
319
320 for k = 1 % Matrices for the conjunct binary variable cases
321 Aordinv(1,10+5*k) = -1;
322 Aordinv(1,5+5*k)  = 1;
323 Aordinv(1,90) = beta;
324 bordinv = -p5(k);
325 end
326
327 for k = 1:2
328 Aordinv1(:,20) = -1;
329 Aordinv1(k,5+5*k)  = 1;
330 Aordinv1(1,90+1) = beta;
331 Aordinv1(2,90+1+(Np-1)) = beta;
332 bordinv1(k,1) = -p5(k);
333 end
334
335 for k = 1:3
336 Aordinv2(:,25) = -1;
337 Aordinv2(k,5+5*k)  = 1;
338 Aordinv2(1,90+2) = beta;
339 Aordinv2(2,90+2+(Np-1)) = beta;
340 Aordinv2(3,90+2+(Np-1)+(Np-2)) = beta;
341 bordinv2(k,1) = -p5(k);
342 end
343
344 for k = 1:4
345 Aordinv3(:,30) = -1;
346 Aordinv3(k,5+5*k)  = 1;
347 Aordinv3(1,90+3) = beta;
348 Aordinv3(2,90+3+(Np-1)) = beta;
```

```
349 Aordinv3(3,90+3+(Np-1)+(Np-2)) = beta;
350 Aordinv3(4,90+3+(Np-1)+(Np-2)+(Np-3)) = beta;
351 bordinv3(k,1) = -p5(k);
352 end
353
354 for k = 1:5
355 Aordinv4(:,35) = -1;
356 Aordinv4(k,5+5*k) = 1;
357 Aordinv4(1,90+4) = beta;
358 Aordinv4(2,90+4+(Np-1)) = beta;
359 Aordinv4(3,90+4+(Np-1)+(Np-2)) = beta;
360 Aordinv4(4,90+4+(Np-1)+(Np-2)+(Np-3)) = beta;
361 Aordinv4(5,90+4+(Np-1)+(Np-2)+(Np-3)+(Np-4)) = beta;
362 bordinv4(k,1) = -p5(k);
363 end
364
365 for k = 1:6
366 Aordinv5(:,40) = -1;
367 Aordinv5(k,5+5*k) = 1;
368 Aordinv5(1,90+5) = beta;
369 Aordinv5(2,90+5+(Np-1)) = beta;
370 Aordinv5(3,90+5+(Np-1)+(Np-2)) = beta;
371 Aordinv5(4,90+5+(Np-1)+(Np-2)+(Np-3)) = beta;
372 Aordinv5(5,90+5+(Np-1)+(Np-2)+(Np-3)+(Np-4)) = beta;
373 Aordinv5(6,90+5+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)) = beta;
374 bordinv5(k,1) = -p5(k);
375 end
376
377 for k = 1:7
378 Aordinv6(:,45) = -1;
379 Aordinv6(k,5+5*k) = 1;
380 Aordinv6(1,90+6) = beta;
381 Aordinv6(2,90+6+(Np-1)) = beta;
382 Aordinv6(3,90+6+(Np-1)+(Np-2)) = beta;
383 Aordinv6(4,90+6+(Np-1)+(Np-2)+(Np-3)) = beta;
384 Aordinv6(5,90+6+(Np-1)+(Np-2)+(Np-3)+(Np-4)) = beta;
385 Aordinv6(6,90+6+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)) = beta;
386 Aordinv6(7,90+6+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)+(Np-6)) = beta;
387 bordinv6(k,1) = -p5(k);
388 end
389
390 Fordtotinv = [Aordinv; Aordinv1; Aordinv2; Aordinv3; Aordinv4;
       Aordinv5; Aordinv6];
391 bordtotinv = [bordinv; bordinv1; bordinv2; bordinv3; bordinv4;
       bordinv5; bordinv6];
392
393 %% Equality constraints w.r.t. the binary variables
```

```
394 Vcon = [1 1 1 1];
395
396 Veq1 = zeros(Np+1, length);
397 beq1 = ones(Np+1, 1);
398
399 for k = 1:(Np+1)
400     Veq1(k, (46+4*k):(49+4*k)) = Vcon; % Constraint one binary
        variable per cycle is chosen
401 end
402
403 Veq2 = zeros(5,length);      % Initialize z(k-1)
404 Veq2(1:5, 1:5) = eye(5);
405 beq2 = zeros(5,1);
406
407 Veq3 = zeros(4,length); % Initialize v(k-1)
408 Veq3(1:4, 46:49) = eye(4);
409 beq3 = zeros(4,1);
410
411 Veq4 = zeros(Np+1,length); % Initialize all inputs equal to 0
412 Veq4(1:(Np+1), (5*(Np+2)+4*(Np+2)+1):(length-sum(1:Np))) = eye(Np
    +1);
413 beq4  = zeros(Np+1, 1);
414
415
416 if t == 0 % Initialize modes for initial scheduler
417 Veq9 = zeros(4,length);
418 for k = 1:4
419 Veq9(k,46+4*k-1+modes(k)) = 1;
420 beq9 = ones(4,1);
421 end
422 end
423
424 %% Building the costfunction
425 C1 = zeros(1,5*(Np+2)); % Cost function for endtimes of all jobs
426 C1(1,5*(Np+2)) = 1;
427 C1(1,5*(Np+2)-5) = 1;
428 C1(1,5*(Np+2)-10) = 1;
429 C1(1,5*(Np+2)-15) = 1;
430 C1(1,5*(Np+2)-20) = 1;
431 C1(1,5*(Np+2)-25) = 1;
432 C1(1,5*(Np+2)-30) = 1;
433 C1(1,5*(Np+2)-35) = 1;
434 C1(1,5*(Np+2)-40) = 1;
435
436 C2 = [zeros(1,4) -0.01*Zvec]; % Cost function for unnecessary mode
    switching
437
```

```
438 C3 = -ones(1,Np+1)*0.01; % Possible cost function for input weight
439
440 C4 = zeros(1, sum(1:Np)); % Cost function for ordering binary
        variables
441
442 C = [C1 C2 C3 C4];
443
444 %% Making the constraint matrices
445 F = [F0; F1; FU; Fordtot; Fordtotinv]; % Inequality constraints
446 b = [b0_0; b1_1; bU; bordtot; bordtotinv];
447
448 if t>=1 % Equality constraints
449 Veq = [Veq1; Veq2; Veq3; Veq4;Veq5;Veq6;Veq7;Veq8];
450 beq = [beq1; beq2; beq3; beq4;beq5;beq6;beq7;beq8];
451
452 else
453     Veq = [Veq1; Veq2; Veq3; Veq4; Veq9];
454     beq = [beq1; beq2; beq3; beq4; beq9];
455 end
456
457 %% Mixed-integer linear programming
458 lb = zeros(length,1);
459 ub = [ones(5*(Np+2),1)*Inf;ones(4*(Np+2),1);ones((Np+1),1)*Inf;
       ones(sum(1:Np), 1)];
460 intcon = [46:81, 90:length];
461
462 [X, FVAL, EXITFLAG] = intlinprog(C', intcon, F, b, Veq, beq, lb, ub
       );
463 X(intcon) = round(X(intcon));
464
465 %% Making the vector Y with all necessary information
466 Y = zeros(10, Np+1);
467
468 for k = 1:(Np+1)
469     Y(1:5,k) = X((6+5*k-5):(5+5*k), :);
470     Y(6:9,k) = X((46+4*k):(49+4*k), :);
471     Y(10, k) = X(81+k, :);
472 end
473
474
475 %% Plotting
476 x = (1:Np+1);
477
478 Z = Y(6:9,:);
479
480
481 for i = 1:(Np+1)
```

```
482 modes(1,i) = find(Z(:,i)>0.1);
483 end
484
485 % figure
486 % plot(x, Y(1:5, :))
487 % legend('x1(k)', 'x2(k)', 'x3(k)', 'x4(k)', 'x5(k)')
488 % title('Starting times of processes per machine')
489 % xlabel('Batch(k)')
490 % ylabel('Starting time(s)')
491 %
492 %
493 % figure
494 % plot(x,Y(6,:), 's', x,Y(7,:), '*', x,Y(8,:), 'o', x,Y(9,:), 'd')
495 % legend('Mode 1', 'Mode 2', 'Mode 3', 'Mode 4')
496 % title('Optimal sequence of modes')
497 % xlabel('Modes')
498 % ylabel('Active or inactive')
499
500 % figure
501 % plot(x, Y(10,:), 'o')
502 % legend('u(k)')
503 % title('Times of incoming batches to process')
504 % xlabel('Batch(k)')
505 % ylabel('Incoming time(s)')
506
507 %% Simulation of begin and endtimes from the optimization
508
509 [Ysimend, Ystart] = dynSimulation(modes, p1, p2, p3, p4, p5, Y, Np)
      ;
510
511 end
```

B-1-3 Scheduler simulation file updating Ystart and Yend

```
1  function [Ysimend, Ystart] = dynSimulation(modes, p1inp, p2inp,
      p3inp, p4inp, p5inp, Y, Np)
2  %%
3  Endtimematrix = zeros(5,(Np+1));
4  Y = Y(1:5,:);
5  for k = 1:Np+1
6  if modes(k) == 1
7      Y(2,k) = 0;
8      Y(4,k) = 0;
9
10 elseif modes(k) == 2
11         Y(2,k) = 0;
12         Y(3,k) = 0;
```

```
13
14 elseif modes(k) == 3
15                     Y(1,k) = 0;
16                     Y(4,k) = 0;
17
18 elseif modes(k) == 4
19                               Y(1,k) = 0;
20                               Y(3,k) = 0;
21 end
22
23 if modes(k) == 1
24 Endtimesmatrix(1,k) = Y(1,k)+p1inp(k);
25 Endtimesmatrix(3,k) = Y(3,k)+p3inp(k);
26 Endtimesmatrix(5,k) = Y(5,k)+p5inp(k);
27
28 elseif modes(k) == 2
29         Endtimesmatrix(1,k) = Y(1,k)+p1inp(k);
30         Endtimesmatrix(4,k) = Y(4,k)+p4inp(k);
31         Endtimesmatrix(5,k) = Y(5,k)+p5inp(k);
32
33 elseif modes(k) == 3
34                     Endtimesmatrix(2,k) = Y(2,k)+p2inp(k);
35                     Endtimesmatrix(3,k) = Y(3,k)+p3inp(k);
36                     Endtimesmatrix(5,k) = Y(5,k)+p5inp(k);
37
38 elseif modes(k) == 4
39                                 Endtimesmatrix(2,k) = Y(2,k)+p2inp(
                                       k);
40                                 Endtimesmatrix(4,k) = Y(4,k)+p4inp(
                                       k);
41                                 Endtimesmatrix(5,k) = Y(5,k)+p5inp(
                                       k);
42 end
43 end
44 Ysimend = Endtimesmatrix;
45 Ystart = Y;
46 end
```

B-1-4 Scheduler simulation file comparison fixed initial scheduler

```
1 function[Ystart, Ysimend, modes, Z, checkFVAL, p1, p2, p3, p4, p5]
     = dynScheduleSim(p1, p2, p3, p4, p5, Np, length, cycle, modes,
     beta, Zvec,X1)
2
3 %% Matrices A0 for mode l=1 cycle k
4 A0_L1 = [0 0 0 0 0;0 0 0 0 0;1 0 0 0 0;0 0 0 0 0; 0 0 1 0 0];
5 A0_L1 = A0_L1-eye(5);
```

```
6  V0_L1 = zeros(5,4);
7  V0_L1(:,1) = [0 0 -beta 0 -beta]';
8
9  F0_L1 = zeros(((Np+1)*5) , length);
10
11 for k = 1:Np+1
12
13 F0_L1((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L1;
14 F0_L1((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
       V0_L1;
15
16 b0_L1((1+5*k-5):(5*k),:) = [-beta -beta -beta-p1(k,:) -beta -beta-
       p3(k,:)]';
17 end
18 %% Matrices A0 for mode l=2 cycle k
19 A0_L2 = [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];
20 A0_L2 = A0_L2-eye(5);
21 V0_L2 = zeros(5,4);
22 V0_L2(:,2) = [0 0 0 -beta -beta]';
23
24 F0_L2 = zeros(((Np+1)*5) , length);
25
26 for k = 1:Np+1
27
28 F0_L2((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L2;
29 F0_L2((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
       V0_L2;
30
31 b0_L2((1+5*k-5):(5*k),:) = [-beta -beta -beta -beta-p1(k,:) -beta-
       p4(k,:)]';
32 end
33 %% Matrices A0 for mode l=3 cycle k
34 A0_L3 = [0 0 0 0 0;0 0 0 0 0;0 1 0 0 0;0 0 0 0 0; 0 0 1 0 0];
35 A0_L3 = A0_L3-eye(5);
36 V0_L3 = zeros(5,4);
37 V0_L3(:,3) = [0 0 -beta 0 -beta]';
38
39 F0_L3 = zeros(((Np+1)*5) , length);
40
41 for k = 1:Np+1
42
43 F0_L3((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L3;
44 F0_L3((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
       V0_L3;
45
46 b0_L3((1+5*k-5):(5*k),:) = [-beta -beta -beta-p2(k,:) -beta -beta-
       p3(k,:)]';
```

```
47 end
48 %% Matrices A0 for mode l=4 cycle k
49 A0_L4 = [0 0 0 0 0;0 0 0 0 0; 0 0 0 0 0;0 1 0 0 0; 0 0 0 1 0];
50 A0_L4 = A0_L4-eye(5);
51 V0_L4 = zeros(5,4);
52 V0_L4(:,4) = [0 0 0 -beta -beta]';
53
54 F0_L4 = zeros(((Np+1)*5) , length);
55
56 for k = 1:Np+1
57
58 F0_L4((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = A0_L4;
59 F0_L4((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k))=
      V0_L4;
60
61 b0_L4((1+5*k-5):(5*k),:) = [-beta -beta -beta -beta-p2(k,:) -beta-
      p4(k,:)]';
62 end
63
64 %% Total constraint matrices for all the modes cycle k
65 F0 = [F0_L1;F0_L2;F0_L3;F0_L4];
66
67 b0_0 = [b0_L1; b0_L2; b0_L3; b0_L4];
68
69 %% Matrices A1 for mode l=1 cycle k
70 A1_L1 = [eye(5) -eye(5)];
71 A1_L1(5,:) = 0;
72 V1_L1 = zeros(5,4);
73 V1_L1(:,1) = -ones(5,1)*beta;
74
75 F1_L1 = zeros((Np+1)*5, length);
76 b1_L1 = zeros((Np+1)*5, 1);
77 for k = 1:Np
78
79 F1_L1((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L1;
80 F1_L1((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
      V1_L1;
81
82 b1_L1((1+5*k-5):(5*k), :) = [-beta-p1(k,:) -beta -beta-p3(k,:) -
      beta -beta]';
83 end
84
85 %% Matrices for A1 for mode l=2 cycle k
86
87 A1_L2 = [eye(5) -eye(5)];
88 A1_L2(5,:) = 0;
89 V1_L2 = zeros(5,4);
```

```
90  V1_L2(:,2) = -ones(5,1)*beta;
91
92  F1_L2 = zeros((Np+1)*5, length);
93  b1_L2 = zeros((Np+1)*5, 1);
94  for k = 1:Np
95
96  F1_L2((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L2;
97  F1_L2((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
        V1_L2;
98
99  b1_L2((1+5*k-5):(5*k), :) = [-beta-p1(k,:) -beta -beta -beta-p4(k
        ,:) -beta]';
100 end
101
102 %% Matrices for A1 for mode l=3 cycle k
103
104 A1_L3 = [eye(5) -eye(5)];
105 A1_L3(5,:) = 0;
106 V1_L3 = zeros(5,4);
107 V1_L3(:,3) = -ones(5,1)*beta;
108
109 F1_L3 = zeros((Np+1)*5, length);
110 b1_L3 = zeros((Np+1)*5, 1);
111 for k = 1:Np
112
113 F1_L3((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L3;
114 F1_L3((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
        V1_L3;
115
116 b1_L3((1+5*k-5):5*k, :) = [-beta -beta-p2(k,:) -beta-p3(k,:) -beta
       -beta]';
117 end
118
119 %% Matrices for A1 for mode l=4 cycle k
120
121 A1_L4 = [eye(5) -eye(5)];
122 A1_L4(5,:) = 0;
123 V1_L4 = zeros(5,4);
124 V1_L4(:,4) = -ones(5,1)*beta;
125
126 F1_L4 = zeros((Np+1)*5, length);
127 b1_L4 = zeros((Np+1)*5, 1);
128 for k = 1:Np
129
130 F1_L4((1+5*k-5):(5*k), (6+5*k-5):(10+5*k)) = A1_L4;
131 F1_L4((1+5*k-5):(5*k), ((5*(Np+2)+1)+4*k):((5*(Np+2)+4)+4*k)) =
        V1_L4;
```

```
132
133 b1_L4((1+5*k-5):5*k, :) = [-beta -beta-p2(k,:) -beta -beta-p4(k,:)
      -beta]';
134 end
135
136 %% Total constraint matrices for all the modes cycle k-1
137 F1 = [F1_L1;F1_L2;F1_L3;F1_L4];
138
139 b1_1 = [b1_L1; b1_L2; b1_L3; b1_L4];
140 %% Constraints for the input mode l=1
141 AU_L1 = -eye(5);
142 VU_L1 = zeros(5,4);
143 VU_L1(:,1) = [-beta 0 0 0 0]';
144 U_L1 = [1 0 0 0 0]';
145
146 FU_L1 = zeros((Np+1)*5, length);
147
148 for k = 1:(Np+1)
149 FU_L1((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L1;
150 FU_L1((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L1;
151 FU_L1((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L1;
152
153 bU_L1((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
154 end
155
156 %% Constraint for the input mode l=2
157 AU_L2 = -eye(5);
158 VU_L2 = zeros(5,4);
159 VU_L2(:,2) = [-beta 0 0 0 0]';
160 U_L2 = [1 0 0 0 0]';
161
162 FU_L2 = zeros((Np+1)*5, length);
163
164 for k = 1:(Np+1)
165 FU_L2((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L2;
166 FU_L2((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L2;
167 FU_L2((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L2;
168
169 bU_L2((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
170 end
171
172 %% Constraints for the input mode l=3
173 AU_L3 = -eye(5);
174 VU_L3 = zeros(5,4);
175 VU_L3(:,3) = [0 -beta 0 0 0]';
176 U_L3 = [0 1 0 0 0]';
177
```

```
178 FU_L3 = zeros((Np+1)*5, length);
179
180 for k = 1:(Np+1)
181 FU_L3((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L3;
182 FU_L3((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L3;
183 FU_L3((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L3;
184
185 bU_L3((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
186 end
187
188
189 %% Constraints for the input mode l=4
190
191 AU_L4 = -eye(5);
192 VU_L4 = zeros(5,4);
193 VU_L4(:,4) = [0 -beta 0 0 0]';
194 U_L4 = [0 1 0 0 0]';
195
196 FU_L4 = zeros((Np+1)*5, length);
197
198 for k = 1:(Np+1)
199 FU_L4((1+5*k-5):(5*k), (6+5*k-5):(5+5*k)) = AU_L4;
200 FU_L4((1+5*k-5):(5*k), (5*(Np+2)+1+4*k):(5*(Np+2)+4+4*k)) = VU_L4;
201 FU_L4((1+5*k-5):(5*k), (5*(Np+2)+4*(Np+2)+k)) = U_L4;
202
203 bU_L4((1+5*k-5):(5*k), 1) = -ones(5,1)*beta;
204 end
205
206
207 %% Total constraint matrices for all input modes at cycle k
208 FU = [FU_L1; FU_L2; FU_L3; FU_L4];
209
210 bU = [bU_L1; bU_L2; bU_L3; bU_L4];
211
212 %% Orderings constraint for the process of machine 5 normal binary
    var
213 Aordk  = zeros(Np, length);
214 Aordk1 = zeros(Np-1, length);
215 Aordk2 = zeros(Np-2, length);
216 Aordk3 = zeros(Np-3, length);
217 Aordk4 = zeros(Np-4, length);
218 Aordk5 = zeros(Np-5, length);
219 Aordk6 = zeros(Np-6, length);
220
221 for k = 1:Np                          % Matrices for the non-
    conjunct cases of the binary variables
222 Aordk(k, 10+5*k) = 1;
```

```
223 Aordk(:,10) = -1;
224 Aordk(k, length-sum(1:Np)+k) = -beta;
225 bordk(k,1) = -beta-p5(k);
226 end
227
228 for k = 2:Np
229 Aordk1(k-1,10+5*k) = 1;
230 Aordk1(:,15) = -1;
231 Aordk1(k-1, length-sum(1:Np)+Np+(k-1)) = -beta;
232 bordk1(k-1,1) =-beta-p5(k);
233 end
234
235 for k = 3:Np
236 Aordk2(k-2,10+5*k) = 1;
237 Aordk2(:,20) = -1;
238 Aordk2(k-2, length-sum(1:Np)+Np+(Np-1)+(k-2)) = -beta;
239 bordk2(k-2,1) =-beta-p5(k);
240 end
241
242 for k = 4:Np
243 Aordk3(k-3,10+5*k) = 1;
244 Aordk3(:,25) = -1;
245 Aordk3(k-3, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(k-3)) = -beta;
246 bordk3(k-3,1) =-beta-p5(k);
247 end
248
249 for k = 5:Np
250 Aordk4(k-4,10+5*k) = 1;
251 Aordk4(:,30) = -1;
252 Aordk4(k-4, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(Np-3)+(k-4)) = -beta
    ;
253 bordk4(k-4,1) =-beta-p5(k);
254 end
255
256 for k = 6:Np
257 Aordk5(k-5,10+5*k) = 1;
258 Aordk5(:,35) = -1;
259 Aordk5(k-5, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(k-5))
    = -beta;
260 bordk5(k-5,1) =-beta-p5(k);
261 end
262
263 for k = 7:Np
264 Aordk6(k-6,10+5*k) = 1;
265 Aordk6(:,40) = -1;
266 Aordk6(k-6, length-sum(1:Np)+Np+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)
    +(k-6)) = -beta;
```

```
267 bordk6(k-6,1) =-beta-p5(k);
268 end
269
270 Fordtot = [Aordk; Aordk1; Aordk2; Aordk3; Aordk4; Aordk5; Aordk6];
271 bordtot = [bordk; bordk1; bordk2; bordk3; bordk4; bordk5; bordk6];
272 %%
273 Aordinv     = zeros(1,length);
274 Aordinv1    = zeros(2,length);
275 Aordinv2    = zeros(3,length);
276 Aordinv3    = zeros(4,length);
277 Aordinv4    = zeros(5,length);
278 Aordinv5    = zeros(6,length);
279
280 for k = 1
281 Aordinv(1,10+5*k) = -1;
282 Aordinv(1,5+5*k)  = 1;
283 Aordinv(1,90) = beta;
284 bordinv = -p5(k);
285 end
286
287 for k = 1:2
288 Aordinv1(:,20) = -1;
289 Aordinv1(k,5+5*k)  = 1;
290 Aordinv1(1,90+1) = beta;
291 Aordinv1(2,90+1+(Np-1)) = beta;
292 bordinv1(k,1) = -p5(k);
293 end
294
295 for k = 1:3
296 Aordinv2(:,25) = -1;
297 Aordinv2(k,5+5*k)  = 1;
298 Aordinv2(1,90+2) = beta;
299 Aordinv2(2,90+2+(Np-1)) = beta;
300 Aordinv2(3,90+2+(Np-1)+(Np-2)) = beta;
301 bordinv2(k,1) = -p5(k);
302 end
303
304 for k = 1:4
305 Aordinv3(:,30) = -1;
306 Aordinv3(k,5+5*k)  = 1;
307 Aordinv3(1,90+3) = beta;
308 Aordinv3(2,90+3+(Np-1)) = beta;
309 Aordinv3(3,90+3+(Np-1)+(Np-2)) = beta;
310 Aordinv3(4,90+3+(Np-1)+(Np-2)+(Np-3)) = beta;
311 bordinv3(k,1) = -p5(k);
312 end
313
```

```
314 for k = 1:5
315 Aordinv4(:,35) = -1;
316 Aordinv4(k,5+5*k)  = 1;
317 Aordinv4(1,90+4) = beta;
318 Aordinv4(2,90+4+(Np-1)) = beta;
319 Aordinv4(3,90+4+(Np-1)+(Np-2)) = beta;
320 Aordinv4(4,90+4+(Np-1)+(Np-2)+(Np-3)) = beta;
321 Aordinv4(5,90+4+(Np-1)+(Np-2)+(Np-3)+(Np-4)) = beta;
322 bordinv4(k,1) = -p5(k);
323 end
324
325 for k = 1:6
326 Aordinv5(:,40) = -1;
327 Aordinv5(k,5+5*k)  = 1;
328 Aordinv5(1,90+5) = beta;
329 Aordinv5(2,90+5+(Np-1)) = beta;
330 Aordinv5(3,90+5+(Np-1)+(Np-2)) = beta;
331 Aordinv5(4,90+5+(Np-1)+(Np-2)+(Np-3)) = beta;
332 Aordinv5(5,90+5+(Np-1)+(Np-2)+(Np-3)+(Np-4)) = beta;
333 Aordinv5(6,90+5+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)) = beta;
334 bordinv5(k,1) = -p5(k);
335 end
336
337 for k = 1:7
338 Aordinv6(:,45) = -1;
339 Aordinv6(k,5+5*k)  = 1;
340 Aordinv6(1,90+6) = beta;
341 Aordinv6(2,90+6+(Np-1)) = beta;
342 Aordinv6(3,90+6+(Np-1)+(Np-2)) = beta;
343 Aordinv6(4,90+6+(Np-1)+(Np-2)+(Np-3)) = beta;
344 Aordinv6(5,90+6+(Np-1)+(Np-2)+(Np-3)+(Np-4)) = beta;
345 Aordinv6(6,90+6+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)) = beta;
346 Aordinv6(7,90+6+(Np-1)+(Np-2)+(Np-3)+(Np-4)+(Np-5)+(Np-6)) = beta;
347 bordinv6(k,1) = -p5(k);
348 end
349
350 Fordtotinv = [Aordinv; Aordinv1; Aordinv2; Aordinv3; Aordinv4;
      Aordinv5; Aordinv6];
351 bordtotinv = [bordinv; bordinv1; bordinv2; bordinv3; bordinv4;
      bordinv5; bordinv6];
352
353 %% Equality constraints w.r.t. the binary variables
354 Vcon = [1 1 1 1];
355
356 Veq1 = zeros(Np+1, length);
357 beq1 = ones(Np+1, 1);
358
```

```
359 for k = 1:(Np+1)
360     Veq1(k, (46+4*k):(49+4*k)) = Vcon;
361 end
362
363 Veq2 = zeros(5,length);
364 Veq2(1:5, 1:5) = eye(5);
365 beq2 = zeros(5,1);
366
367 Veq3 = zeros(4,length);
368 Veq3(1:4, 46:49) = eye(4);
369 beq3 = zeros(4,1);
370
371 Veq4 = zeros(Np+1,length);
372 Veq4(1:(Np+1), 82:(length-sum(1:Np))) = eye(Np+1);
373 beq4  = zeros(Np+1, 1);
374
375 %%
376 Veq5 = zeros(4*(Np+2)+sum(1:Np),length);
377 beq5 = zeros(4*(Np+2)+sum(1:Np),1);
378
379 Veq5(1:4*(Np+2), 46:81) = eye(36);
380 Veq5(4*(Np+2)+1:(4*(Np+2)+sum(1:Np)), 90:117) = eye(28);
381
382 beq5(1:(4*(Np+2)),1) = X1(46:81);
383 beq5(4*(Np+2)+1:(4*(Np+2)+sum(1:Np)),1) = X1(90:117);
384
385 %% Building the costfunction
386 C1 = zeros(1,5*(Np+2));
387 C1(1,5*(Np+2)) = 1;
388 C1(1,5*(Np+2)-5) = 1;
389 C1(1,5*(Np+2)-10) = 1;
390 C1(1,5*(Np+2)-15) = 1;
391 C1(1,5*(Np+2)-20) = 1;
392 C1(1,5*(Np+2)-25) = 1;
393 C1(1,5*(Np+2)-30) = 1;
394 C1(1,5*(Np+2)-35) = 1;
395 C1(1,5*(Np+2)-40) = 1;
396
397 C2 = [zeros(1,4) -0.01*Zvec'];
398
399 C3 = -ones(1,Np+1)*0.01; %Weight matrix of the input
400 %C3 = zeros(1,Np+1)*0.0001; %Weight matrix of the input
401 C4 = zeros(1, sum(1:Np));
402
403 C = [C1 C2 C3 C4];
404
405 %% Making the constraint matrices
```

```
406 F = [F0; F1; FU;Fordtot; Fordtotinv];
407 b = [b0_0; b1_1; bU; bordtot; bordtotinv];
408
409 Veq = [Veq1; Veq2; Veq3; Veq4; Veq5];
410 beq = [beq1; beq2; beq3; beq4; beq5];
411
412
413 %% Mixed-integer linear programming
414 lb = zeros(length,1);
415 ub = [ones(5*(Np+2),1)*Inf;ones(4*(Np+2),1);ones((Np+1),1)*Inf;
       ones(sum(1:Np), 1)];
416 intcon = [46:81, 90:length];
417
418 [X, FVAL, EXITFLAG] = intlinprog(C', intcon, F, b, Veq, beq, lb, ub
       );
419
420 checkFVAL = FVAL;
421
422 %% Making the vector Y with all necessary information
423 Y = zeros(10, Np+1);
424
425 for k = 1:(Np+1)
426     Y(1:5,k) = X((6+5*k-5):(5+5*k), :);
427     Y(6:9,k) = X((46+4*k):(49+4*k), :);
428     Y(10, k) = X(81+k, :);
429 end
430 %% Plotting
431 x = (1:Np+1);
432
433 Z = Y(6:9,:);
434
435 for i = 1:(Np+1)
436 modes(1,i) = find(Z(:,i)>0.1);
437 end
438
439 % figure
440 % plot(x, Y(1:5, :))
441 % legend('x1(k)', 'x2(k)', 'x3(k)', 'x4(k)', 'x5(k)')
442 % title('Starting times of processes per machine')
443 % xlabel('Batch(k)')
444 % ylabel('Starting time(s)')
445 %
446 %
447 % figure
448 % plot(x,Y(6,:), 's', x,Y(7,:), '*', x,Y(8,:), 'o', x,Y(9,:), 'd')
449 % legend('Mode 1', 'Mode 2', 'Mode 3', 'Mode 4')
450 % title('Optimal sequence of modes')
```

```
% xlabel('Modes')
% ylabel('Active or inactive')

% figure
% plot(x, Y(10,:), 'o')
% legend('u(k)')
% title('Times of incoming batches to process')
% xlabel('Batch(k)')
% ylabel('Incoming time(s)')

%% Simulation of the real starting times
[Ysimend, Ystart] = dynSimulation(modes, p1, p2, p3, p4, p5, Y, Np)
    ;
end
```

B-2 Subsystems MATLAB-files

B-2-1 Mashing main file

```
function [p1, xlim, xlimest] = mashing1(t_inp, seed)
%% Data gathering for mashing tank 1

Ts = 0.01;
a_a = 2;
k_0 = 0.011497;
t_0 = 0;
t_fin = 1000;
x0 = [100;0;0];
zeta = 0.99995;
a = k_0/3;
Q = a^2;
R = 0.1;

[x1tank1, x2tank1, x3tank1, ymash1, n, t, w, v] = datamashing(Ts,
    a_a, k_0, t_0, t_fin, x0, zeta, Q, R, seed);

%% Estimation for mashing tank 1
PInit = eye(3);
xInit = [100;0;0];

[xest] = EKFmashing(PInit, xInit, ymash1, Ts, a_a, k_0, zeta, Q, R,
     t_fin);

% figure
% plot(t(1:100*t_fin), xest(:,1), t(1:100*t_fin), x1tank1(2:100*
    t_fin+1))
```

```
25 % legend('x1est', 'x1real')
26 %
27 %
28 % figure
29 % plot(t(1:100*t_fin), xest(:,2), t(1:100*t_fin), x2tank1(2:100*
       t_fin+1))
30 % legend('x2est', 'x2real')
31 %
32 %
33 % figure
34 % plot(t(1:100*t_fin), xest(:,3), t(1:100*t_fin), x3tank1(2:100*
       t_fin+1))
35 % legend('x3est', 'x3real')
36
37 xlim = find(x2tank1>80);
38 xlim = xlim(1);
39
40 xlimest = find(xest(:,2)>80);
41 xlimest = xlimest(1);
42
43 %% Prediction at time t for endtime
44 x0 = xest(t_inp*1000+1, :)';
45 xpred1(1) = x0(1);
46 xpred2(1) = x0(2);
47 xpred3(1) = x0(3);
48
49 %%
50
51 for k = 1:(t_fin*100-(1000*t_inp))
52
53 xpred = f(xpred1(k), xpred2(k), xpred3(k), Ts, a_a, k_0, zeta);
54 xpred1(k+1) = xpred(1);
55 xpred2(k+1) = xpred(2);
56 xpred3(k+1) = xpred(3);
57 end
58
59 xnew = [xest(1:(1000*t_inp)+1,2); xpred2(2:end)'];
60
61 % figure
62 % a = size(xnew);
63 % a = a(1);
64 % plot(1:a, xnew)
65 % hold on
66 % plot(1:a, x2tank1)
67 % xlabel('Time(mDay)')
68 % ylabel('Percentage of sugars created(%)')
69 % legend('xPred', 'xReal')
```

```
70
71
72 xpredest = find(xnew>80);
73 p1 = round((xpredest(1)/1000), 1);
74 end
```

B-2-2 Data gathering mashing function

```
1  function [x1, x2, x3, y, n, t, w, v] = datamashing(Ts,a_a, k_0, t_0
       , t_fin, x0, zeta, Q, R, seed)
2  x1(1) = x0(1);
3  x2(1) = x0(2);
4  x3(1) = x0(3);
5  t(1) = 0;
6
7  n = (t_fin-t_0)/Ts;
8  w = sqrt(Q)*wgn(n,1,0, [], 100*seed+1);
9  v = sqrt(R)*wgn(n,1,0, [], seed);
10
11 for k = 1:n
12     x1(k+1) = x1(k)-Ts*(a_a*(k_0+x3(k))*x1(k));
13     x2(k+1) = x2(k)+Ts*(a_a*(k_0+x3(k))*x1(k));
14     x3(k+1) = zeta*x3(k)+ Ts*w(k);
15
16     t(k+1) = k;
17     y(k) = x1(k)+v(k);
18 end
19 %
20 % figure
21 % subplot(3,1,1)
22 % plot(t,x1);
23 % subplot(3,1,2)
24 % plot(t,x2);
25 % subplot(3,1,3)
26 % plot(t,x3);
27
28
29 % figure
30 % subplot(2,1,1)
31 % plot(t,x3)
32 % subplot(2,1,2)
33 % plot(t(1:length(t)-1), w)
34 end
```

B-2-3 EKF mashing

```
function [xest] = EKFmashing(PInit, xInit, y, Ts, a_a, k_0, zeta, Q
    , R, t_fin)

H = [0;0;Ts];
C = [1 0 0];


%% Initialization
Pnextnext = PInit;

x1(1) = xInit(1);
x2(1) = xInit(2);
x3(1) = xInit(3);

%%
for k = 1:(t_fin*100-1)

Pnextprev = df(x1(k), x2(k),x3(k), Ts, a_a, k_0, zeta)*Pnextnext*(
    df(x1(k),x2(k),x3(k), Ts, a_a, k_0, zeta)')+H*Q*H';

xhat = f(x1(k), x2(k), x3(k), Ts, a_a, k_0, zeta);

K = (Pnextprev*C')/(C*Pnextprev*C'+ R);

Pnextnext = (eye(3)- K*C)*Pnextprev;

x = xhat+K*(y(k+1)-C*xhat);

x1(k+1) = x(1);
x2(k+1) = x(2);
x3(k+1) = x(3);
end

xest =[x1;x2;x3];
xest = xest';
end
```

B-2-4 Function for state-space

```
function [x] = f(x1,x2,x3, Ts, a_a, k_0, zeta)

x = [x1 - Ts*a_a*k_0*x1-Ts*a_a*x3*x1; x2 + Ts*a_a*k_0*x1+Ts*a_a*x3*
    x1; zeta*x3];

end
```

B-2-5 Function derivative state-space

```
function [x] = df(x1,x2,x3,Ts, a_a, k_0, zeta)

x = [1-Ts*a_a*k_0-Ts*a_a*x3 0 -Ts*a_a*x1;Ts*a_a*k_0+Ts*a_a*x3 1 Ts*
    a_a*x1;0 0 zeta];

end
```

B-2-6 Brewing main file

```
function [p3, xlim, xlimest] = brewing1(t_inp, seed)
%% Data gathering for brewing tank 1
Ts = 0.01;
b_0 = 0.026829;
t_0 = 0;
t_fin = 1000;
x0 = [100;0;0];
zeta = 0.99996;
a = b_0/3;
Q = a^2;
R = 0.1;

[x1brew, x2brew, x3brew, ybrew, n, t, w, v] = databrewing(Ts, b_0,
    t_0, t_fin, x0, zeta, Q, R, seed);

%% Estimation for brewing tank 1
PInit = eye(3);
xInit = [100;0;0];

[xestbrew] = EKFbrewing(PInit, xInit, ybrew, Ts, b_0, zeta, Q, R,
    t_fin);
%
% figure
% plot(t(1:(end-1)), xestbrew(:,1), t(1:(end-1)), x1brew(2:end))
% legend('xest1', 'x1real')
%
%
%
% figure
% plot(t(1:(end-1)), xestbrew(:,2), t(1:(end-1)), x2brew(2:end))
% legend('xest2', 'x2real')
%
%
% figure
% plot(t(1:(end-1)), xestbrew(:,3), t(1:(end-1)), x3brew(2:end))
% legend('x3est', 'x3real')

xlim = find(x2brew>80);
```

```
37 xlim = xlim(1);
38
39 xlimest = find(xestbrew(:,2)>80);
40 xlimest = xlimest(1);
41 %% Prediction
42 x0 = xestbrew(t_inp*1000+1, :)';
43 xpredbrew1(1) = x0(1);
44 xpredbrew2(1) = x0(2);
45 xpredbrew3(1) = x0(3);
46
47
48 %%
49 for k = 1:(t_fin*100-(1000*t_inp))
50
51 xpredbrew = fbrew(xpredbrew1(k), xpredbrew2(k), xpredbrew3(k),Ts,
      b_0, zeta);
52
53 xpredbrew1(k+1) = xpredbrew(1);
54 xpredbrew2(k+1) = xpredbrew(2);
55 xpredbrew3(k+1) = xpredbrew(3);
56 end
57
58 xnewbrew = [xestbrew(1:(t_inp*1000)+1,2); xpredbrew2(2:end)'];
59
60 % figure
61 % a = size(xnew);
62 % a = a(1);
63 % plot(1:a, xnew)
64 % hold on
65 % plot(1:a, x2tank1)
66 % xlabel('Time(mDay)')
67 % ylabel('Percentage of sugars created(%)')
68 % legend('xPred', 'xReal')
69
70
71 xpredestbrew = find(xnewbrew>80);
72 p3 = round((xpredestbrew(1)/1000), 1);
73
74 end
```

B-2-7 Data gathering brewing function file

```
1 function [x1, x2, x3, y, n, t, w, v] = databrewing(Ts, b_0, t_0,
      t_fin, x0, zeta, Q, R, seed)
2 x1(1) = x0(1);
3 x2(1) = x0(2);
4 x3(1) = x0(3);
```

```
5  t(1) = 0;
6
7  n = (t_fin-t_0)/Ts;
8  w = sqrt(Q)*wgn(n,1,0, [], 100*seed+1);
9  v = sqrt(R)*wgn(n,1,0, [], seed);
10
11 for k = 1:n
12     x1(k+1) = x1(k)-Ts*b_0*x1(k)-Ts*x3(k)*x1(k);
13     x2(k+1) = x2(k)+Ts*b_0*x1(k)+Ts*x3(k)*x1(k);
14     x3(k+1) = zeta*x3(k)+ Ts*w(k);
15
16     t(k+1) = k;
17     y(k) = x1(k)+v(k);
18 end
19 % figure
20 % subplot(3,1,1)
21 % plot(t,x1);
22 % subplot(3,1,2)
23 % plot(t,x2);
24 % subplot(3,1,3)
25 % plot(t,x3);
26
27 % figure
28 % subplot(2,1,1)
29 % plot(t,x3)
30 % subplot(2,1,2)
31 % plot(t(1:length(t)-1), w)
32 end
```

B-2-8 EKF brewing

```
1  function [xest] = EKFbrewing(PInit, xInit, y, Ts, b_0, zeta, Q, R,
       t_fin)
2
3  C = [1 0 0];
4  H = [0; 0; Ts];
5
6  Pnextnext = PInit;
7
8  x1(1) = xInit(1);
9  x2(1) = xInit(2);
10 x3(1) = xInit(3);
11
12 %%
13
14 for k = 1:(100*t_fin-1)
15
```

```
16 Pnextprev = dfbrew(x1(k), x2(k),x3(k), Ts, b_0, zeta)*Pnextnext*(
       dfbrew(x1(k),x2(k),x3(k), Ts, b_0, zeta)')+H*Q*H';
17
18 xhat = fbrew(x1(k), x2(k), x3(k), Ts, b_0, zeta);
19
20 K = (Pnextprev*C')/(C*Pnextprev*C'+ R);
21
22 Pnextnext = (eye(3)- K*C)*Pnextprev;
23
24 x = xhat+K*(y(k+1)-C*xhat);
25
26 x1(k+1) = x(1);
27 x2(k+1) = x(2);
28 x3(k+1) = x(3);
29 end
30
31 xest =[x1;x2;x3];
32 xest = xest';
33
34
35
36
37
38 end
```

B-2-9 Function for state-space

```
1 function [x] = fbrew(x1,x2,x3, Ts, b_0, zeta)
2
3 x = [x1-Ts*b_0*x1-Ts*x3*x1; x2+Ts*b_0*x1+Ts*x3*x1;zeta*x3];
4
5 end
```

B-2-10 Function derivative state-space

```
1 function [x] = dfbrew(x1,x2,x3, Ts, b_0, zeta)
2
3 x = [1-Ts*b_0-Ts*x3 0 -Ts*x1;Ts*b_0+Ts*x3 1 Ts*x1; 0 0 zeta];
4
5
6 end
```

Bibliography

[1] F. (Francois) Baccelli. *Synchronization and linearity : an algebra for discrete event systems*. Wiley, 1992.

[2] Chris A Brackley, David S Broomhead, M Carmen Romano, and Marco Thiel. A max-plus model of ribosome dynamics during mrna translation. *Journal of Theoretical Biology*, 303:128–140, 2012.

[3] Cédric Brandam, XM Meyer, J Proth, P Strehaiano, and Hervé Pingaud. An original kinetic model for the enzymatic hydrolysis of starch during mashing. *Biochemical Engineering Journal*, 13(1):43–52, 2003.

[4] Charles K Chui, Guanrong Chen, et al. *Kalman filtering*. Springer, 2017.

[5] B de Andrés-Toro, JM Girón-Sierra, JA Lopez-Orozco, C Fernandez-Conde, José Martínez Peinado, and F Garcıa-Ochoa. A kinetic model for beer production under industrial operational conditions. *Mathematics and Computers in Simulation*, 48(1):65–74, 1998.

[6] Bart De Schutter and Ton Van Den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, 2001.

[7] Bart De Schutter, Ton van den Boom, Jia Xu, and Samira S Farahani. Analysis and control of max-plus linear discrete-event systems: An introduction. *Discrete Event Dynamic Systems*, 30(1):25–54, 2020.

[8] Bernd F Heidergott. *Max-plus linear stochastic systems and perturbation analysis*, volume 15. Springer Science & Business Media, 2006.

[9] Jan Komenda, Sébastien Lahaye, J-L Boimond, and Ton van den Boom. Max-plus algebra in the history of discrete event systems. *Annual Reviews in Control*, 45:240–249, 2018.

[10] Gabriel AD Lopes, Bart Kersbergen, Bart De Schutter, Ton van den Boom, and Robert Babuška. Synchronization of a class of cyclic discrete-event systems describing legged locomotion. *Discrete Event Dynamic Systems*, 26(2):225–261, 2016.

[11] Mark G Malowicki and Thomas H Shellhammer. Isomerization and degradation kinetics of hop (humulus lupulus) acids in a model wort-boiling system. *Journal of agricultural and food chemistry*, 53(11):4434–4439, 2005.

[12] A Seleim and H ElMaraghy. Max-plus modeling of manufacturing flow lines. *Procedia CIRP*, 17:71–75, 2014.

[13] FB Van Boetzelaer, Ton JJ van den Boom, and RR Negenborn. Model predictive scheduling for container terminals. *IFAC Proceedings Volumes*, 47(3):5091–5096, 2014.

[14] Ton van den Boom and Bart De Schutter. Stabilizing model predictive controllers for randomly switching max-plus-linear systems. In *2007 European Control Conference (ECC)*, pages 4952–4959. IEEE, 2007.

[15] Ton JJ van den Boom, Hilco de Bruijn, Bart De Schutter, and Leyla Özkan. The interaction between scheduling and control of semi-cyclic hybrid systems. *IFAC-PapersOnLine*, 51(7):212–217, 2018.

[16] Ton JJ van den Boom and Bart De Schutter. Modelling and control of discrete event systems using switching max-plus-linear systems. *Control engineering practice*, 14(10):1199–1211, 2006.

[17] Ton JJ van den Boom and Bart De Schutter. Modeling and control of switching max-plus-linear systems with random and deterministic switching. *Discrete Event Dynamic Systems*, 22(3):293–332, 2012.

[18] Ton JJ van den Boom, Marenne van den Muijsenberg, and Bart De Schutter. Model predictive scheduling of semi-cyclic discrete-event systems using switching max-plus linear models and dynamic graphs. *Discrete Event Dynamic Systems*, 30(4):635–669, 2020.

[19] Ronnie Willaert. The beer brewing process: Wort production and beer. *Handbook of food products manufacturing*, 2:443, 2007.