Joint Advanced Student School 2008, St. Petersburg

# Optimization of Hybrid Control Systems in Manufacturing

Final Report by

cand. ing. Michael Fall

born on 23.05.1981
address:
Nymphenburger Strasse 121
80636 München
Tel.: 089 50003174

Institute of
AUTOMATIC CONTROL ENGINEERING
Technische Universität München

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Sandra Hirche
Univ.-Prof. Dr.-Ing. Olaf Stursberg

Supervisor: Dipl.-Ing. Tina Paschedag
Submitted: 07.04.2008

# Contents

# Chapter 1

# Introduction

This report summarizes the research activity on optimal control problems for hybrid systems in manufacturing mainly proposed by C. G. Cassandras and D. L. Pepyne [Pep00]. In the first chapter an overview of control problems related to manufacturing is given. The need to develop a hybrid system framework for a solution of the optimal control problem is explained and some basics about hybrid systems in general are proposed. The second chapter explains the hybrid system framework in more detail and shows several ways on how to interpret the optimal control problem. What follows is the analysis of the optimization problem and a discussion of possible solutions. The chapter ends with a presentation of the algorithm specifically tailored to the introduced hybrid system framework.

## 1.1 Control Problems in Manufacturing

Considering the manufacturing process of a metal-making company individual metal-ingots undergo various operations during production (e.g rolling, milling, machining metals) until the final product is released.
A common operation in metal-making involves slowly heating up the metal-strips in an oven, keeping them at a certain temperature level and cooling them down again under control (annealing) to achieve a desired grain structure (stiffness, hardness). In this case the task of the process control is to determine when to switch operation times in order to achieve a defined heating profile.
This brief description of the production process shows already the major control problems that are related to manufacturing. A number of time-consuming individual steps need to be processed to achieve a high *quality* in the finished product. When trying to integrate these processes into the plant-wide scheduling of the production plant this objective is clearly in conflict with the importance of a *timely delivery* of the products.
In manufacturing it is a common problem to find a control strategy that is able to deal with these trade-offs between quality aspects and job completion deadlines. The hybrid system framework introduced in [Pep00] is able to cope with these conflicts

and to find a solution on the proposed optimal control problem.

## 1.2 A Hybrid System Framework for Manufacturing

To formulate the optimal control problem mathematically a suitable process model is required that takes into account all aspects of the manufacturing process. As the process model shall be applicable to any manufacturing process of the same type a general description of the individual subtasks is required. Therefor all process action that actually changes the physical characteristics of a product is simply referred to as *jobs*. The abstract term *server* is used to characterize all workstations (e.g. an oven) each job has to visit during manufacturing.

During processing there is a change in the physical characteristics (shape, quality) of each job. These *time-driven dynamics* can be described by a set of (continuous) differential equations. Within the production scheme processing start and stop times represent *discrete-event* dynamics and they are modeled by equations derived form queuing networks theory.
As discrete and continuous dynamics are interacting very closely together they define a *hybrid* system for the manufacturing process.

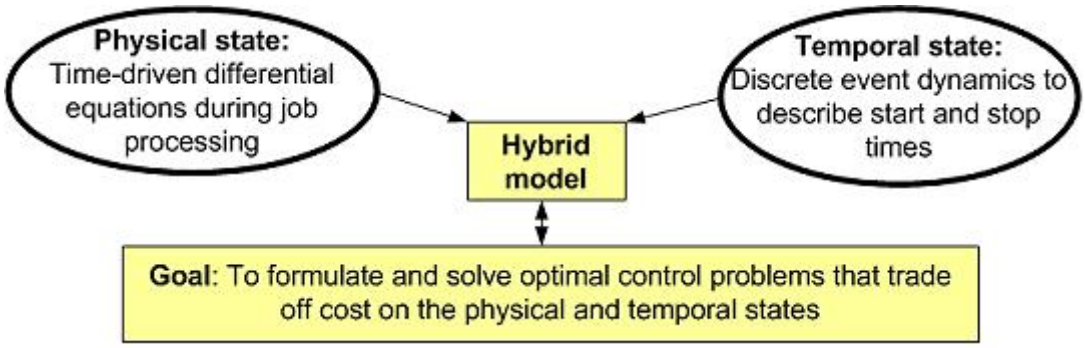


Figure 1.1: Physical and temporal states within a hybrid system

The framework proposed to solve the optimal control problem refers to these forms of the dynamics as the *physical states* (e.g temperature, energy) and the *temporal states* (operation start and stop times).

## 1.3 General remarks on Hybrid Systems

Hybrid systems are used in general for processes that combine event-driven with time-driven dynamics behavior. A simple example for a hybrid system is a climate

control system used to keep the room temperature within a certain interval (figure 1.2). Similar examples for hybrid systems may be found in [Stu07].

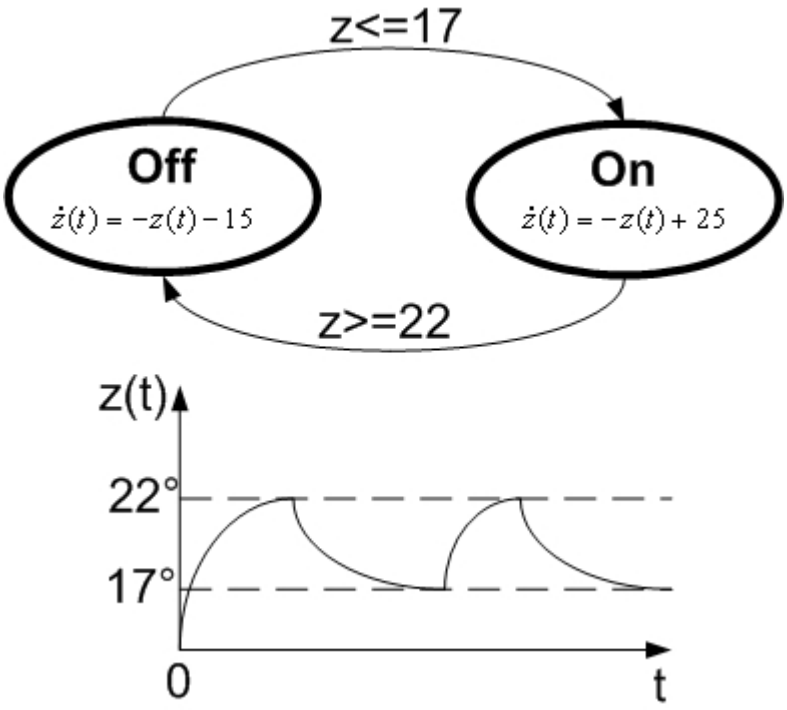


Figure 1.2: A climate control systems as an example for a hybrid system

The discrete states are simply defined as $Q = 0, 1$ which refers to the radiator being in "On" or "Off"-state. The transitions between these two states are depending on the continuous variable (= room temperature). During each state the room temperature evolves according to some time-driven dynamics. This movement of the physical state trajectories is shown in the second chart of figure 1.2.

In general there are various types of modeling framework for hybrid systems. The climate control systems may be described using a *hybrid automaton*. Some other model framework extend time-driven models that allow discrete events to be injected. Again others are based on event-driven models with an extension of time-driven activities between event occurrences.

Referring back to the manufacturing processes, the hybrid systems framework can be viewed as an extension of a *queuing network model*. This framework will be explained in more detail in the following chapter.

# Chapter 2

# Optimal Control of Hybrid Systems in Manufacturing

## 2.1 Modeling of a Single-Stage Manufacturing Process

### 2.1.1 A Single-Server Queuing System

For the following analysis and solution of the optimal control problem, the manufacturing process considered is limited to a single operation. A suitable model to represent this process is shown in figure 2.1 as a *single-server queuing system*.

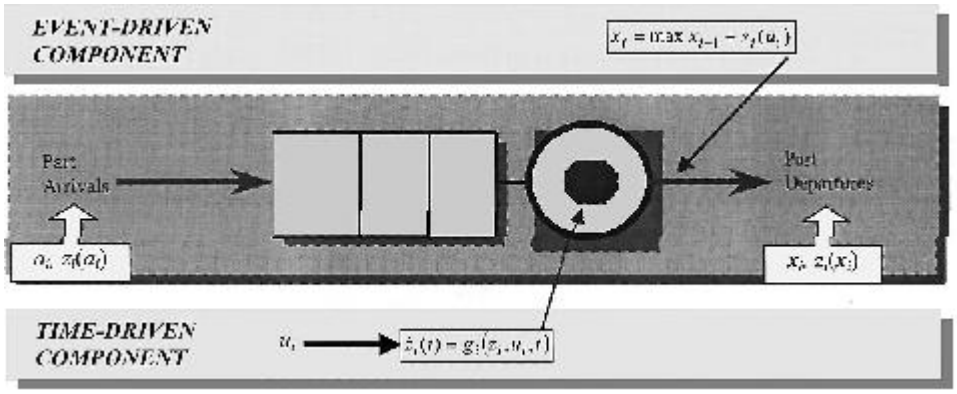


Figure 2.1: Modeling of a single-stage manufacturing process

As in standard queuing theory the buffer or waiting area is represented as a set of (open) rectangles. Incoming jobs are stored in this buffer ("queue") until they are processed by the server. The server itself is denoted by a circle and is subjected to several restrictions: It never idles when there are jobs in the queue and service is never interrupted once processing on a job has started ("non-preemptive server"). The entire scheme of job processing follows a first-come, first-served principle (FCFS).

A series of $i = 1, 2, ..., N$ jobs is assigned to the server by an external resource. Job arrival times $0 \leq a_1 \leq a_2 \leq ... \leq a_N \leq \infty$ are assumed to be known. The processing time for each single job is denoted by $s_i(u_i)$ where $u_i$ is the control variable. A further restriction is that the control $u_i$ is considered as being *time-invariant* over the course of the processing time $s_i$.
The system is *hybrid* in the sense that it combines continuous (time-driven) dynamics with discrete (event-driven) dynamics:

1. *Time-Driven-Dynamics:* During job processing the physical state $z_i$ of a job $i$ changes according to a deterministic differential equation:

$$\dot{z_i}(t) = g_i(z_i, u_i, t); \qquad z_i(\tau_i) = \zeta_i \tag{2.1}$$

   where $\tau_i$ is the processing start time for job $i$ and $\zeta_i$ is the state of the job at that time.

2. *Event-Driven Dynamics:* The standard Lindley equation for a FCFS server is used to describe the completion time $x_i$ for each job:

$$x_i = \tau_i + s_i(u_i) = max(x_{i-1}, a_i) + s_i(u_i) \tag{2.2}$$

   where $a_i$ is the processing start time of job $i$.

Justifying the hybrid nature of the system, equations 2.1 and 2.2 show that the control $u_i$ can affect both, time-driven and event-driven dynamics.

### 2.1.2 Control Policy for a Hybrid System Framework

The control policy for the hybrid system framework is to determine how the jobs are being processed through the system optimally. Since process arrival times are assumed to be known the goal is to adjust the processing time for each individual job so that production requirements are met optimally.
Considering the optimal control problem in a more general way one can distinguish between several technical sub-problems related to a manufacturing process:

1. Compute control trajectories for optimally steering the physical system state.
2. Choose the optimal processing time for each job.
3. Determine the order of job processing.
4. Consider the sequence of servers each job has to visit.

For each individual subproblem there exist several solutions on who to come up with a solution. A solution on 1) is given through *Nonlinear Optimal Control*. 2) can be tackled by taking into account *discrete-event dynamic system performance* wheres

*scheduling methods* provide some useful approaches on how to solve subproblem 3) and 4).
The problem of a hybrid system is that within such a system all of those four subproblems are coupled very tightly together and a "general" solution on how to solve the optimal control problem is extremely hard to find. The crucial point is that in general available methods collapse by taking into account all of those four subproblems.

### 2.1.3 Interpretation of the Hybrid System Framework

Referring back to the single-server queuing system introduced in section 2.1.1 an appropriate way on how to interpret the queuing mechanism might be as a discrete event system that is extended by some time driven dynamics. The time-driven dynamics represent the physical state of each job and the discrete events are determined by the temporal state.

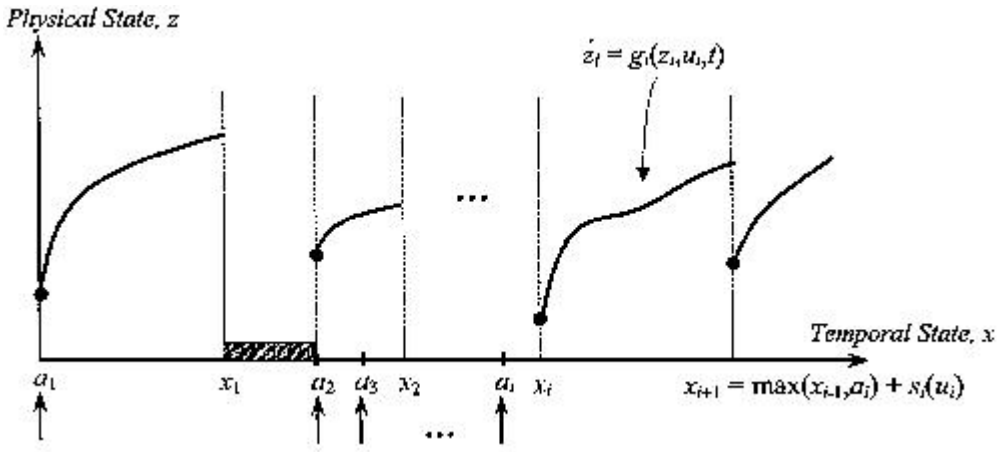


Figure 2.2: Typical state trajectories of a hybrid system

Figure 2.2 shows the evolution of the physical state $z_i$ as a function of the temporal state (which simply can be interpreted as the "time-axis"). The continuous dynamics are "interrupted" by exogenous (uncontrolled) job arrival events $a_i$ and by controlled job release events $x_i$.
Consider job 1: After the uncontrolled job arrival event $a_1$ processing on the job starts and its physical state begins to move according to $\dot{z}_1 = g_1(z, u_1, t)$. At the time $x_1$ job 1 is released from the server and as there are no further jobs in the queue the server runs in idle-mode waiting for the arrival of new jobs.

The processing time $s_i(u_i)$ for each job has to be chosen with respect to a certain quality "level" denoted by $\Gamma_i$ each job must fulfill:

$$s_i(u_i) = min[t \geq 0 : z_i(\tau_i + t) = \int_{\tau_i}^{\tau_i + t} g_i(s, u_i, t)ds + \zeta_i \in \Gamma_i] \tag{2.3}$$

Equation 2.3 is referred to as the "stopping rule" and it means that it is only possible to release a job from the server after its physical state $z_i$ has reached a level that is above the desired quality level $\Gamma_i$.

## 2.2 Formulation of the Optimal Control Problem

Regarding the optimization problem introduced by the hybrid system framework there exist at least two optimization goals that are clearly in conflict with each other. On the one hand the processing time for each job has to be chosen in a way that we do not miss any job completion deadlines (upper bound for the processing time). On the other hand it is desirable to guarantee a certain quality level which defines the minimum processing time for each job (lower bound for the processing time). To match both of these requirements the hybrid system framework needs to trade-off time critical issues against quality aspects.
As for any optimization problem a crucible step is to set up a suitable cost function $J$ that takes into account the desired behavior of the system. For the above defined hybrid system framework the optimal control objective is to choose a control policy $\pi = u_1, u_2, ..., u_N$ to minimize an objective cost function of the form:

$$\min_{\pi} J = \sum_{i=1}^{N} \Theta_i(u_i) + \Psi_i(x_i) \tag{2.4}$$

where $\Theta_i$ is a function to assign a cost on the control $u_i$ and $\Psi_i(x_i)$ is used for charging delayed job completion times $x_i$. Equation 2.4 shows that there is no explicit cost on the physical state $z_i$. But clearly the stopping criterion (equation 2.3) guarantees that the physical state of each completed job satisfies the pre-defined quality objectives $z_i(x_i) \in \Gamma_i$.
By taking a closer look on the cost function two different ways on how to interpret the optimal control problem can be identified.

### 2.2.1 Class-1 Control Problems

For the first class of control problems the cost function $J(\Theta_i, \Psi_i)$ trades off quality aspects against job completion times and the control $u_i$ is simply interpreted as the processing time $s_i$. Mathematically Class-1 problems need to satisfy the following requirements for each job $i = 1, ..., N$ :

1. $\Theta_i(.)$ is twice continuously differentiable, strictly convex and monotonically decreasing.

2. $\Psi_i(i)$ is twice continuously differentiable, strictly convex and its minimum is obtained for a finite point $\delta_i$.

3. $s_i(.)$ is linear with $s_i(u_i) = \alpha \cdot u_i$

For this type of control problems the physical state $z_i$ is interpreted as a measure of the achieved quality. If the processing time $s_i$ is chosen beyond a certain point there are increasing costs due to a lack of quality in the manufacturing process.

### 2.2.2 Class-2 Control Problems

This type of control problems trades off the processing speed against the job completion time by an appropriate choice of the cost function $J(\Theta_i, \Psi_i)$. The control $u_i$ is interpreted as the effort applied to the job. An example for the abstract term "effort" might be the energy that is applied to a furnace to achieve a defined heating profile. The following rules hold for each job:

1. $\Theta_i(.)$ is twice continuously differentiable, strictly convex and monotonically increasing.

2. $\Psi_i(i)$ is twice continuously differentiable, strictly convex and monotonically increasing.

3. $s_i(.)$ is twice continuously differentiable, strictly convex and monotonically decreasing.

When considering a class-2 problem the goal is to process each job until it has met a defined final state $z_i(x_i) = q$ starting from an initial raw state $z_i(\tau_i) = \zeta_i = 0$. As the control $u_i$ is interpreted as the effort applied to each job it is convenient to interpret the movement of the physical state for each job simply as the control $\dot{z}_i = u_i$.

An example for a class-2 control problem is given in [Pep00]:

$$s_i(u_i) = \tfrac{q}{u_i} \tag{2.5}$$

$$\Theta_i(u_i) = u_i^2 \tag{2.6}$$

$$\Psi_i(x_i) = \begin{cases} 0, & if \quad x_i < \delta_i \\ (x_i - \delta_i)^2, & if \quad x_i \geq \delta_i \end{cases} \tag{2.7}$$

The processing time $s_i$ in equation 2.7 results from the desired quality level and the effort used to reach this level. To satisfy the mathematical definitions of a class-2 control problem a typical choice for $\Theta_i(u_i)$ is to assign a quadratic cost on the effort applied to each job. To penalize long processing times beyond a certain due date again a quadratic approach for $\Psi_i(x_i)$ is utilized.

## 2.3 Analysis of the Optimization Problem

For the analysis of class-1 optimization problems a more general form of the cost function is discussed in [PW01]. In this case $J(x_i, \lambda, u_i,)$ is not split-up into the sub-functions of $\Theta_i(.)$ and $\Psi_i(.)$ but is regarded as one function in $x_i, \lambda, u_i$.

$$J(x, \lambda, u) = \sum_{i=1}^{N} L_i(x_i, u_i) + \lambda_i[(x_{i-1}, a_i) + s_i(u_i) - x_i] \tag{2.8}$$

where $a_i$ and $x_i$ are again job processing start- and stop times and $\lambda_i$ is an N-dimensional vector for the co-state. A necessary condition for a point to be a local minimum requires the first partial derivatives of equation 2.8 to be zero. This leads to the following expressions:

$$\frac{\partial J}{\partial u_i} = 0 \quad \Rightarrow \quad \frac{\partial L_i(x_i, u_i)}{\partial u_i} + \lambda_i \frac{ds_i(u_i)}{du_i} = 0 \tag{2.9}$$

$$\frac{\partial J}{\partial \lambda_i} = 0 \quad \Rightarrow \quad x_i = max(a_i, x_i) + s_i(u_i) \tag{2.10}$$

$$\frac{\partial J}{\partial x_i} = 0 \quad \Rightarrow \quad \lambda_i = \frac{\partial L(x_i, u_i)}{\partial x_i} + \lambda_{i+1} \frac{d \cdot max(x_i, a_{i+1})}{d \cdot x_i} \tag{2.11}$$

Equations 2.10-2.11 define a *two-point-boundary-value problem* (TPBVP) which cannot be solved using standard algorithms. The fundamental problem is the nondifferentiability of the *max*-function in the co-state equation (2.11). This function is clearly not defined at the point where its arguments are equal ($x_i = a_{i+1}$). At all other points it is differentiable with:

$$\frac{d}{dx_i} max(x_i, a_{i+1}) = \begin{cases} 0, & if \quad x_i < a_{i+1} \\ 1, & if \quad x_i > a_{i+1} \end{cases} \tag{2.12}$$

This special structure of the cost function introduced by the even-generating mechanism of the hybrid system framework makes it impossible to use basic variational (gradient-based) methods. Those methods require the cost function to be "smooth" (differentiable) everywhere.
Another method of solution might be given through dynamic programming (DP) which uses algorithms based on recursion and memorization. The problem about DP is that the computational effort to search over the whole policy space for the optimal solution is extremely high even for modest optimization problem. For the purpose of the above defined hybrid system framework this means that the number of jobs considered has to be fairly small.
Furthermore first-order approximations might be considered when trying to solve the optimal control problem. Such solutions reduce the complexity of computation significantly but always yield the danger to end up in a local- instead of a global extremum.

A more general solution that takes into account the nondifferentiability of the *max*-function is introduced in the next section that explains some of the basic ideas of nonsmooth optimization theory.

### 2.3.1 Nonsmooth Optimization

To illustrate the problem of a nonsmooth cost function in more detail an example for a class-1 optimization problem with a number of jobs $N = 2$ is considered. The following example may be found in [PW01].

$$\Theta_1(u_1) = \frac{1}{u_1}, \qquad \Theta_2(u_2) = \frac{1}{u_2} \tag{2.13}$$

$$\Psi_1(x_1) = x_1^2, \qquad \Psi_2(x_2) = (x_2 - 30)^2 \tag{2.14}$$

By evaluating equation 2.4 for known job arrival times $a_1 = 2$ and $a_2 = 3$ the cost function $J(u_1, u_2)$ results to:

$$J(u_1, u_2) = \frac{1}{u_1} + \frac{1}{u_2} + (2 + u_1)^2 + [max(2 + u_1, 3) + u_2 - 30]^2 \tag{2.15}$$

where the last term in equation 2.15 includes the *max*-function that holds the problem of the nondifferentiability. Figure 2.3 clearly shows that the cost function is in general not smooth. The points of nondifferentiabily are located along the crease

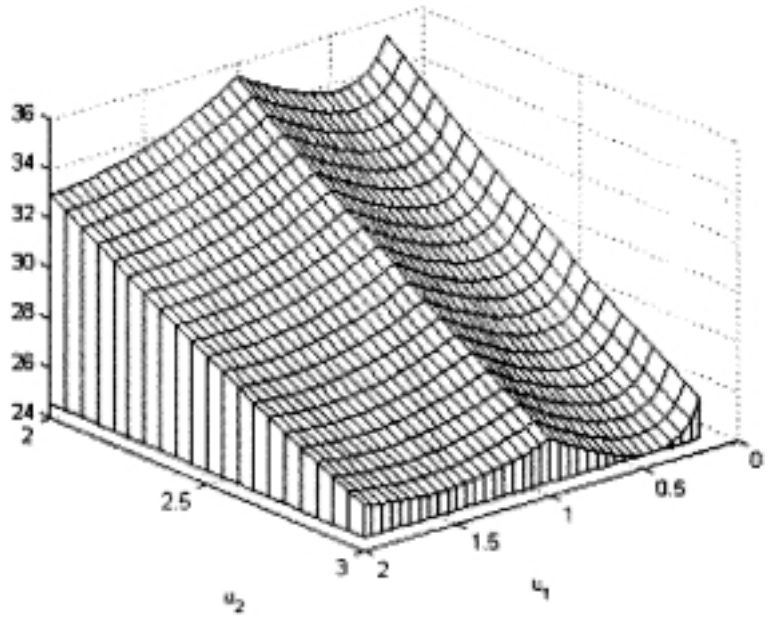


Figure 2.3: Plot of a portion of $J(u_1, u_2)$

running in the direction of $u_2$. At those points job arrival times of the second job

$a_2$ coincide with the release times of job one ($x_1$). Those jobs remark a critical component within the solution and the goal is to exclude them from computation. In general they are defined as:

**Definition**: A job $i = 1, ..., N$ is called *critical* if $x_i = a_{i+1}$.

where $x_i$ is again the release time for the active job and and $a_{i+1}$ is the arrival time of the next job.

Critical jobs reflect the idea of a *just-in-time* production scheme which means that the processing time $s_i(u_i)$ is chosen such that a job is processed as long as possible and that is released from the server at the earliest with the arrival of the next job.

In order to come up with a solution for a nonsmooth cost function the *maximum-principle concept* introduced in equations 2.10-2.11 needs to be extended by the definition of Lipschitz continuous functions. For a Lipschitz continuous function the inequality defined in equation 2.16 has to be satisfied.

$$|f(x) - f(y)| \leq K|x - y| \tag{2.16}$$

where $K$ is a positive constant and $x, y \in E$ are an open subset of $\Re^N$. Lipschitz functions are continuous but need not be differentiable everywhere. The objective function in equation 2.4 satisfies those requirements and is a Lipschitz continuous one.

Due to the event-generating mechanism cost functions in hybrid optimal control are normally Lipschitz continuous and the procedure of finding the optimal solution is a generalized one.
When considering continuously differentiable functions a necessary condition for a point to be a local extremum is that the gradient is zero at that point. Since for Lipschitz continuous functions the gradient does not exist everywhere, necessary conditions for optimality are formulated as a *generalization of the gradient* [Pep00]. This generalization makes use of the *subdiffertial* $f$ at $u$, denoted by $\partial f(u)$.

The most important property of the subdifferential is that if $u$ is a local extremum of $f$ then $0 \in \partial f(0)$. This behavior of the subdifferential provides a method to check if the optimal solution for a nonsmooth optimization problem fulfills the conditions for optimality. Solving the optimization problem therefor requires deriving an expression for the subdifferential $J(u_1, ..., u_N)$.

### 2.3.2 Subdifferential Derivation

As an example for the evaluation of the subdifferential the function $f(x) = |x|$ is considered. This function is Lipschitz continuous since it is differentiable everywhere except at the origin.

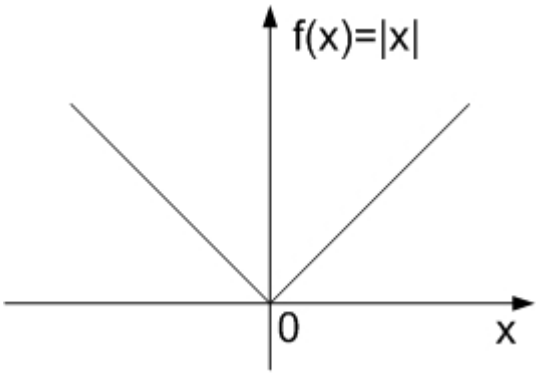


Figure 2.4: Plot of $f(x) = |x|$

For the evaluation of the subdifferential the left and right derivative of $f(x)$ at the origin are given through:

$$\lim_{x \uparrow 0-} (df(x)/dx) = -1 \tag{2.17}$$

$$\lim_{x \uparrow 0+} (df(x)/dx) = +1 \tag{2.18}$$

Then the subdifferential $\partial f(u)$ is defined as $\partial f(0) = [-1, 1]$ which represents just a closed interval on the real-axis and clearly $0 \in \partial f(0)$.

For the control algorithm presented in section 2.4 the evaluation of the subdifferential is not much different than from the one presented by the *norm-of-x* function. When calculating the left and right derivative of possible job release times a simple sign check is implemented in the code to evaluate the subdifferential.

### 2.3.3 Definiton of a Sample Path

When making use of the subdifferential it is useful to introduce a *sample path* which is defined by a sequence of job arrival- and job release times. Such a sample path can be divided into *busy periods* and *idle periods*. During a busy period the server is actively processing on a job and when the server is in idle mode then there are no further jobs in the queue that need to be processed. Figure 2.5 shows a sequence of such a sample path.
By taking a closer look on the sample path in figure 2.5 a block structure can be identified which allows to split up a busy period into several sub-blocks. Such a sub-block is determined by the timely occurrence of critical jobs. Each sub-block (except the last one in in the busy period) is terminated by a critical job.
For example consider job $(m + 1)$: Since $x_m = a_{m+1}$ this job is a critical one and therefor determines the end of the first block. The second block within the busy period starts with the first job after the critical one.

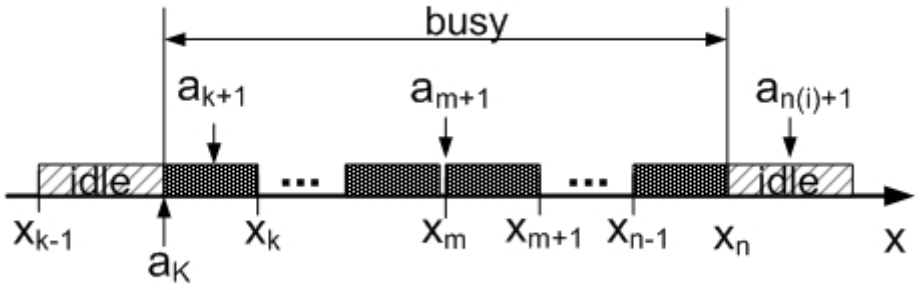


Figure 2.5: Separation of a sample path into busy and idle periods

Referring back to the subdifferential which is now given through the left and right derivative of $J(u_1, ..., u_N)$ with respect to $u_1, ..., u_N$ consider a control scheme that is limited to a single block with jobs $j = i, ..., m(i)$. It is possible to adjust the departure times for each job through:

$$x_j = max(x_{i-1}, a_i) + s_i(u_i) + \sum_{k=i+1}^{j} s_k(u_k) \tag{2.19}$$

Since for all jobs within the block (except the last one) $x_{m(i)} < a_{m(i)+1}$ the *max*-function is clearly defined. For the last job in the block the left and right derivatives are evaluated with respect to the control $u_i$:

$$\zeta_i^- = \lim_{x_{m(i)\uparrow a_{m(i)+1}}} \frac{\partial J}{\partial u_i} \tag{2.20}$$

$$\zeta_i^+ = \lim_{x_{m(i)\downarrow a_{m(i)+1}}} \frac{\partial J}{\partial u_i} \tag{2.21}$$

Using these equations (2.21 and 2.21) is now possible to prove whether a solution for the optimal control problem is a unique one. According to [GW98] the inequality $\zeta_i^- \leq \zeta_i^+$ must be true for all jobs $i = 1, ..., N$ which leads to the subdifferential of the objective function:

$$\partial J = [\zeta_1^-, \zeta_1^+] \times ... \times [\zeta_N^-, \zeta_N^+] \subset \Re^N \tag{2.22}$$

Now the optimal control sequence for each job $i = 1, ..., N$ must satisfy $0 \in [\zeta_i^-, \zeta_i^+] \subset \Re^1$.

### 2.3.4 Decoupling Properties

The decoupling properties of the sample path allow the decomposition of the optimal state trajectory into fully independent sub-segments which simplify the analysis of the optimization problem. The following statements for the decoupling properties are a summary of the lemmas established in [Pep00], proofs for them can be found in [GW98].

**Idle Period Decoupling Property**

- The optimal controls $u_i^*$ are only dependent on job arrival times and on the number of jobs within each busy period.
- The controls $u_i$ for individual busy periods can be computed independently.

**Block related Decoupling Property**

- Controls $u_i$ for jobs before and and after a critical job are independent of each other.
- Each busy period can be split-up into blocks which can be calculated independently.

The idea of these decoupling properties is to solve the large optimization problem as a series of smaller independent subproblems. This allows to restrict the number of degrees of freedom for each subproblem and speeds up computation significantly. An essential step when making use of these properties is the identification of the busy period structure.

### 2.3.5 Critical Job Identification

It can be shown that for the hybrid system framework almost any sample path will have critical jobs and therefore will contain points of nondifferentiabliy in the objective function. This section explains some of the main properties that can be used to identify the busy period structure.

Without loss of generality the first busy period of a sample with a number of jobs $i = 1, ..., B$ is considered. The optimal job departure times are denoted by $x_{i_B}, i = 1, ..., B$ where $i$ is the job index and $B$ is the number of jobs within the busy period. It can be shown [PW01] that for such a busy period:

- The optimal job departure times $x_{i,B}$ are only dependent on $a_1$ and $B$.
- The optimal job departure times are monotonically decreasing with $B$.

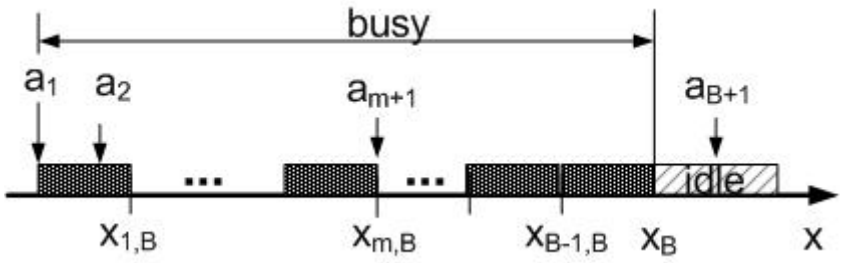


Figure 2.6: A busy period with precomputed optimal job departure times

Making use of these properties a pre-computation of the optimal job departure times for all jobs within the busy period is possible. Figure 2.6 shows such a busy period with given arrival times $a_i$ and optimal departure times $x_{i,B}$.
Since intervals of the form $[x_{i,B}, x_{i,i}]$ for any $B$ and $i = 1, ..., B-1$ are now well defined (because $x_{i,B} \leq x_{i,i}$), it is possible to make a statement whether or not a busy period will contain any critical job.

> **Lemma 1**: A busy period with jobs $i = 1, ..., B$ contains at least one critical job if $a_{i+1} \in [x_{i,B}, x_{i,i}]$ for one or more jobs $i = 1, ..., B-1$.

Such an interval $[x_{i,B}, x_{i,i}]$ is referred to as a *critical interval*. To identify which of these jobs is critical the following lemma is introduced in [Pep00].

> **Lemma 2**: Considering a busy period with a number of $N$ jobs remaining to be processed: If there exists some $L \leq N$ such that $a_{i+1} < x_{i,N}$ and $x_{L,L+1} \leq a_{L+1} \leq x_{L,L}$ then job $L$ is critical.

In other words Lemma 2 means that depending on job arrival times and the pre-computed optimal departure times, one can make a statement whether or not a job is critical.


Figure 2.7: Critical intervals for an example with N=3 (considering job 1)

As an example for the critical job identification figure 2.7 shows a sample paths for a number of three jobs. The optimal job departure times $x_{1,1}, x_{1,2}, ..., x_{3,3}$ are

precomputed for a given arrival time of the first job $a_1$. When applying the introduced lemmas to the sample path, the location of $a_2$ relative to the critical interval $[x_{1,2}, x_{1,1}]$ allows to determine whether job 1 is critical, whether it ends the first busy period or whether it is part of the busy period containing at least job 1 and job 2.


Figure 2.8: Critical intervals for an example with N=3 (considering job 1-3)

By considering the last sample path in figure 2.8 the need for a further criterion on the identification of critical jobs arises. If $x_{1,3} \leq a_2 \leq x_{1,2}$ and $x_{2,3} \leq a_3 \leq x_{2,2}$ then both jobs satisfy the condition for a critical one and it is not possible to determine which of them or if both of them are critical.

In this case equation 2.22 helps to find a unique solution. Each critical job must satisfy $0 \in [\zeta_i^-, \zeta_i^+]$. Therefor by checking the sign of the left and right derivatives for each job it can be shown which one of these jobs are critical.

## 2.4 A Backward Recursive Algorithm

This section shall give a brief overview of the algorithm presented in [Pep00] to find a solution on the optimal control problem. The essential idea of the algorithm is to decompose the overall nonsmooth optimization problem into a series of smooth nonlinear subproblems that can be solved using standard gradient-based solvers (TPBVP). Such algorithms are presented in [Kir70]. By doing so the algorithm invented by C. G. Cassandras and D. L. Pepyne makes use of *terminal constraints* (TCs).

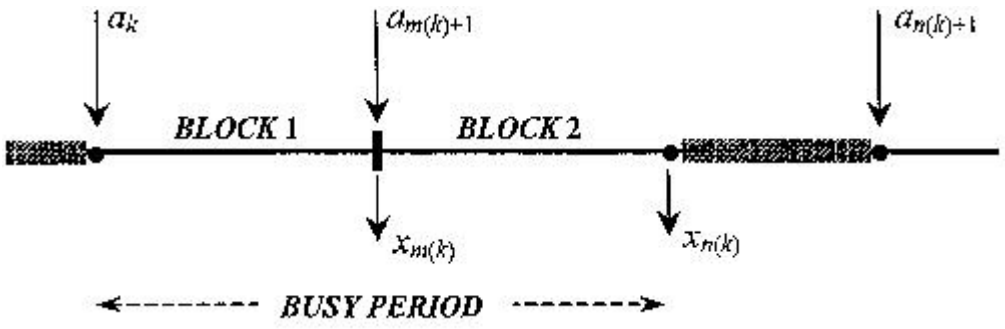


Figure 2.9: A busy period consisting of tho blocks

Figure 2.9 shows a busy period containing jobs $k, ..., n(k)$ which is subdivided in two blocks. Then by the idle period decoupling property and the partial decoupling property introduced in section 2.3.4 the optimal controls for the jobs in the busy period can be determined by solving two independent TPBVPs. A TC $x_{m(k)} = a_{m(k)+1}$ is used to determine the end of the first block. The second block does not end with a TC since this is the last block in the busy period and therefor can never contain any critical job.

These terminal constraints provide a useful way on how to determine the block structure of a sample path. A definition of each individual sub-problem is given in equation 2.23:

$$Problem \quad P_{j,k}(C): \quad \min_{u_j,...u_k} J = \sum_{i=j}^{k} \Theta_i(u_i) + \Psi_i(x_i) \tag{2.23}$$

where, if $C = 1$ a terminal constraint $x_k = a_{k+1}$ is added to the block and for the case $C = 0$ no critical jobs are considered and the problem solver is called through $P_{j,k}(0)$. Furthermore the following definition for the derivatives is given:

$$\zeta_{j,k} = \frac{d\Theta_j}{du_j} + \frac{ds_j}{du_j} \sum_{i=j}^{k} \frac{d\Psi_i}{dx_i} \tag{2.24}$$

respectively for the left and right derivatives of jobs $N = j, ..., k$.

The algorithm operates in a *backward-recursive* manner: Calculation starts with the last job $N$ in the queue for which the optimal control $u_N^*$ and the optimal job release time $x_N^*$ is determined. Next, the last but one job $N - 1$ is considered. To determine the optimal controls $u_{N-1}^*, u_N^*$ and the optimal release times $x_{N-1}^*, x_N^*$ it is first assumed that the job is a critical one.
Then by computing the quantities $\zeta_{4,4}$ and $\zeta_{4,5}$ according to equation 2.24 a statement about the coupling between these two jobs can formulated. A simple sign test performed on these quantities tells about whether this job should go into the same busy period or whether it should start a new one. This procedure is executed on each further job and it finishes with the computation of the first job in the queue (backward-recursive).

For a more detailed description of the algorithm including some example code the reader is referred to [CP00] and [Pep99].

# Chapter 3

# Summary

This report gives an overview of the hybrid system framework introduced by C. G. Cassandras and D. L. Pepyne designed to deal with optimal control problems related to manufacturing. As a restriction only single-stage processes are analyzed by the author although the framework could be easily extended towards multi-stage processes.
The frameworks´ main objective is to trade-off time / quality aspects, both conflicting issues by an appropriate choice of the cost function. Two ways on how to interpret the optimal control problem are presented. Depending on the role of the control - either as the service time for an individual job - or as the effort applied to a job the cost function is chosen differently.

The analysis of the optimization problem shows the difficulty in finding a solution because of nondifferentiability and nonconvexity of the objective function. Nonsmooth optimization with Lipschitz continuous functions is introduced to deal with these properties of the cost function.

The solution approach proposed in [Pep00] follows a "divide-and-conquer" scheme that decomposes the large-scale nonsmooth optimization problem into a series of smaller-scale smooth subproblems. For each of these subproblems fast numerical algorithms can be used for computation. Crucial for this solution approach is a precise identification of the physical structure of the sample path. The algorithm presented in the last section consequently makes use of the decoupling properties stated in [PW01]. It proceeds in a backward-recursive manner starting with the last job in the queue and adding each previous job one by one.

The presented method systematically takes advantage of the structural properties of hybrid systems. An intelligent analysis of the optimal control problem allows to avoid the burden of finding a "general" solution for the hybrid system framework which is instead solved as a series of independent subproblems.

# List of Figures

# Bibliography

[CP00] Y. Wardi C. G. Cassandras and D. L. Pepyne. *Algorithm for computing optimal controls for single-stage hybrid manufacturing systems.* submitted for publication, 2000.

[GW98] M. Gazarik and Y. Wardi. *Optimal release times in a single server: An optimal control perspective.* IEEE Transactions on Automatic Control, Vol. 43, pp. 998-1002, 1998.

[Kir70] D. E. Kirk. *Optimal Control Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1970.

[Pep99] D. L. Pepyne. *Performance optimization strategies for discrete event and hybrid systems.* Ph.D. Dissertation, Dept. of Electrical an Computer Engineering, Univ. of Massachusetts, Amherst, 1999.

[Pep00] C. G. Cassandras D. L. Pepyne. *Optimal Control of Hybrid Systems in Manufacturing.* Proceedings of the IEEE, Vol. 88, No.7, 2000.

[PW01] C. G. Cassandras D. L. Pepyne and Y. Wardi. *Optimal control of a class of hybrid systems, submitted for publication.* IEEE Transactions on Automatic Control, 2001.

[Stu07] O. Stursberg. *Script on lecture "Discrete Event and Hybrid Systems".* Technical University of Munich, 2007.