Optimale Ablaufplanung für zyklische Prozesse mit Pooling-Ressourcen

Optimal Scheduling of Cyclic Processes with Pooling-Resources

Eckart Mayer, Kai Wulff, Christoph Horst und Jörg Raisch

Dieser Beitrag behandelt die zeitoptimale Ablaufplanung für zyklische Prozesse mit Pooling-Ressourcen. Wir stellen einen Ansatz vor, mit dem eine große Klasse zyklischer Ablaufplanungsprobleme modelliert werden kann. Wir zeigen, wie das resultierende Modell durch Reduktion der Freiheitsgrade und Reparametrierung vereinfacht werden kann, ohne die optimale Lösung auszuschließen. Durch eine geeignete Transformation wird das reduzierte Problem auf ein lineares gemischt-ganzzahliges Optimierungsproblem abgebildet, das mit etablierten Verfahren effizient gelöst werden kann. Pooling-Ressourcen, d. h. Ressourcen, die bei jeder Aktivierung eine vorgegebene Anzahl von Werkstücken/Proben verarbeiten müssen, lassen sich ebenfalls in den vorgestellten Ansatz integrieren.

The topic of this contribution is the time-optimal scheduling of cyclic processes with pooling resources. We present a modelling framework that captures a large class of cyclic scheduling problems. We show how the resulting model can be considerably simplified without cutting the optimal solution. This involves a reduction and a reparametrisation of the degrees of freedom. The reduced model can be transformed into a linear mixed-integer program (MILP), for which a globally optimal solution can be efficiently computed using standard algorithms and tools. We also show how pooling resources can be treated in our framework. Pooling resources have a capacity greater than one and must, at each activation, be loaded to capacity.

Schlagwörter: Ablaufplanung, gemischt-ganzzahlige Optimierung, zyklische Systeme, Pooling

Keywords: Scheduling, mixed-integer programming, cyclic systems, pooling

1 Einleitung

Viele Systeme in Natur und Technik besitzen zyklischen Charakter. Sie werden durch die zeitliche Abfolge von Ereignissen beschrieben, die sich periodisch über eine große (oder beliebige) Zahl von Zyklen hinweg wiederholen. In technischen Systemen kann Zyklizität entweder eine intrinsische Forderung sein (weil beispielsweise Arbeitsschritte immer im selben Zeitmuster erfolgen müssen) oder aber vom Anwender vorgegeben werden, um das Systemverhalten geeignet zu vereinfachen.

Das wichtigste Merkmal eines zyklischen Systems ist seine *Zyklus*- bzw. *Taktzeit*, also die Dauer des einzelnen Zyklus bzw. der Zeitversatz zwischen dem Start aufeinander folgender Zyklen. Der Kehrwert der Taktzeit liefert den Durchsatz des Systems. Ziel der meisten Anwendungen ist die Maximierung des Durchsatzes, also die Minimierung der Zykluszeit.

Die optimale Ablaufplanung (Scheduling, z. B. [2; 14]) für zyklisch betriebene Systeme befasst sich mit der optimalen zeitlichen Zuordnung von Aktivitäten zu beschränkten Ressourcen unter Zyklizitäts-Nebenbedingungen. Solche Probleme treten in einer Vielzahl von Anwendungsgebieten auf, beispielsweise in der Verfahrenstechnik (z. B. [1]), bei der Prozessorbelegung in Parallelrechensystemen (z. B. [7]), bei flexiblen Fertigungssystemen (z. B. [4]) oder im Be-

reich des High-Throughput Screening (HTS) [13]. Formale Lösungsansätze für zyklische Scheduling-Probleme finden sich in [3; 5; 9; 16; 17]; dort werden aber einschränkendere Annahmen getroffen als in der vorliegenden Arbeit (siehe Abschnitt 2).

Wir stellen eine Methode vor, mit der eine große Klasse zyklischer Scheduling-Probleme global optimal gelöst werden kann. Sie basiert auf einem ereignisdiskreten Systemmodell, das in einem ersten Schritt mit Verfahren der Graphentheorie vereinfacht werden kann. Durch eine effiziente Formulierung der Ausschlussbedingungen (disjunctive constraints) lässt sich das reduzierte zyklische Scheduling-Problem als gemischt-ganzzahlige nichtlineare Optimierungsaufgabe (mixed-integer nonlinear program, MINLP) schreiben. Letztere kann schließlich durch eine geeignete Transformation *exakt* in ein *lineares* gemischt-ganzzahliges Problem (mixed-integer linear program, MILP) überführt werden. Dieses lässt sich dann mit etablierten Verfahren und Werkzeugen effizient global optimal lösen (z. B. [10]).

Eine wesentliche Neuerung der vorliegenden Arbeit ist die Einbeziehung sogenannter Pooling-Ressourcen in die dargestellte formale Vorgehensweise. Solche in vielen Anwendungsgebieten auftretenden Ressourcen müssen bei jeder Aktivierung eine genau festgelegte Zahl von Werkstücken/Proben verarbeiten [5; 15]. Beispiele sind Öfen, die nur voll beladen beheizt werden oder Zentrifugen mit Aufnahmeplätzen für mehrere Proben, die zur Vermeidung von Unwucht nur voll bestückt betrieben werden dürfen. Man beachte, dass die mathematische Behandlung von Pooling-Ressourcen sich deutlich von der Behandlung sog. paralleler Ressourcen (Mehrkapazitäts-Ressourcen) unterscheidet. Letztere können mehrere Werkstücke/Proben unabhängig voneinander bearbeiten [12].

Der Aufsatz ist folgendermaßen gegliedert: In Abschnitt 2 werden zyklische Systeme eingeführt und die an sie gestellten Anforderungen diskutiert. Abschnitt 3 beschäftigt sich mit der Modellbildung, Abschnitt 4 mit der Modellreduktion und der Formulierung der Ausschlussbedingungen (disjunctive constraints) mit Hilfe endlich vieler ganzzahliger Variablen. Abschnitt 5 beschreibt, wie sich das resultierende MINLP durch eine geeignete Transformation exakt in ein MILP überführen lässt. Abschnitt 6 befasst sich mit der Erweiterung des vorgestellten Ansatzes auf Pooling-Ressourcen. Die Beispiele in Abschnitt 5 und 6 besitzen in erster Linie illustrativen Charakter. Die beschriebenen Verfahren wurden aber in Zusammenarbeit mit industriellen Kooperationspartnern auch auf weit komplexere Probleme erfolgreich angewandt.

2 Zyklische Systeme

Ein zyklischer Ablauf ist dadurch charakterisiert, dass ein generischer Satz von Arbeitsschritten für eine große bzw. beliebige Anzahl von Elementen in immer gleicher Weise durchgeführt wird. Eine Instanz dieses Satzes von Arbeitsschritten – oder *Operationen* – wird im Rahmen dieses Beitrags als *Batch* bezeichnet.

Tabelle 1: Notation.

T	Taktzeit
m	Anzahl der Ressourcen
n	Anzahl der Aktivitäten im Batch-Zeitschema
J_i	Ressource für Aktivität i
$\mathcal{X}$	Menge aller Ereignisse im Batch-Zeitschema
$t\{x\}$	Zeitpunkt von Ereignis x im Batch-Zeitschema
$t\{x^{(\rho)}\}$	Zeitpunkt von Ereignis x in Batch ρ
$t\{o_i\}$	Zeitpunkt der Ressourcenbelegung durch Aktivität i
$t\{r_i\}$	Zeitpunkt der Ressourcenfreigabe durch Aktivität i
$z_{(i1,i2)}$	ganzzahlige Variable für Ausschlussbedingung
θ_k	Zeiten zur Parametrierung der Ereigniszeitpunkte im Batch-Zeitschema
K	Anzahl der Zeitvariablen θ_k
ι_{max}	Anzahl der Aktivitätenpaare $(i1, i2)$ im Batch-Zeitschema mit $J_{i1} = J_{i2}$ und $i1 < i2$
C	Poolingkapazität
T^*	innere Zykluszeit bei Pooling

Das untersuchte System sei in eine Anzahl von *Ressourcen* unterteilt, mit der Eigenschaft, dass jede Ressource maximal von einem Batch gleichzeitig belegt sein kann.[1] Ein Batch umfasst dann eine Menge von nicht unterbrechbaren Belegungen jeweils einer dieser Ressourcen (*Aktivitäten*).

Im Allgemeinen können mehrere Aktivitäten eines Batches gleichzeitig stattfinden (auf verschiedenen Ressourcen). Ein Batch kann mehrere zu verarbeitende Einheiten (*Werkstücke*) umfassen[2], aber auch bereits ein einzelnes Werkstück kann mehrere Ressourcen gleichzeitig belegen (z. B. eine Transporteinheit und einen Fahrwegabschnitt).

Beispiel: Die Ressourcen einer Produktionsanlage im Sinne dieser Definition sind Maschinen, Transporteinheiten, Fahrwegabschnitte oder Lagerplätze. Ein Batch umfasst dann die Menge aller Aktivitäten, d. h. Ressourcenbelegungen, die zur Herstellung eines Exemplars des Endprodukts erforderlich sind. Dabei können verschiedene Werkstücke beteiligt sein, die zwischendurch zerlegt, parallel auf verschiedenen Maschinen weiterverarbeitet werden und schließlich zu einem Endprodukt montiert werden. Mit derselben Struktur kann der zyklische Betrieb von Screeninganlagen beschrieben werden. Dabei fasst ein Batch die zum Screening einer Probe erforderlichen Aktivitäten zusammen (einschließlich paralleler Operationen wie der Zuführung von Reagenzien etc.)

[1] Diese Definition wird in Abschnitt 6 auf sogenannte Pooling-Ressourcen erweitert.
[2] Der Begriff Werkstück wird im Folgenden in verallgemeinerter Bedeutung verwendet und kann je nach Anwendung auch für eine Probe (z. B. beim HTS), ein Stück Software (bei Parallelrechnersystemen) oder Ähnliches stehen.

Die Problemstruktur wird durch folgende Eigenschaften charakterisiert:

1. Jeder Aktivität ist eindeutig eine Ressource zugeordnet. Umgekehrt benötigt der einzelne Batch jedoch möglicherweise dieselbe Ressource mehrmals.
2. Aktivitäten beinhalten einen oder mehrere Arbeitsschritte/Operationen (von deterministischer Dauer). Sie werden durch Ereignisse gegliedert und beinhalten unter anderem je ein Start- und Endereignis.
3. Zwischen den Ereignissen des Batches bestehen zeitliche Abhängigkeiten – innerhalb von Aktivitäten oder zwischen Ereignissen in verschiedenen Aktivitäten desselben Batches. Damit sind die Aktivitäten eines Batches teilweise voneinander abhängig, teilweise jedoch auch unabhängig voneinander parallel ausführbar (sog. Präzedenznetzwerk).
4. Die zeitlichen Abhängigkeiten zwischen den Ereignissen können neben Mindestdauern auch die Form *oberer Zeitschranken* besitzen. Solche Nebenbedingungen treten beispielsweise häufig bei verfahrenstechnischen Anwendungen auf.
5. Zwischen den Ressourcen existieren oft keine Zwischenspeicher. In diesem Fall kann eine Aktivität frühestens enden, wenn eine Nachfolgeaktivität für das Werkstück auf der nächsten benötigten Ressource begonnen hat.

Die Punkte 3 bis 5 stellen eine weitreichende Verallgemeinerung gegenüber Job-Shop-Problemen [2] oder sequentiell angeordneten Fertigungsstraßen (Flow Shop) dar. Diese allgemeine Definition ermöglicht es unter anderem auch, die Dauer einer Aktivität künstlich zu verlängern, soweit dies nicht gegen obere Zeitschranken verstößt. Dies ist beispielsweise dann notwendig, wenn die für das Werkstück anschließend benötigte Ressource noch belegt ist und kein Zwischenspeicher existiert (Punkt 5).

Zyklische Abläufe sind dadurch gekennzeichnet, dass das System von einer (beliebig) großen Zahl der beschriebenen Batches in exakt identischem zeitlichen Muster durchlaufen wird. Darüber hinaus besteht zwischen aufeinander folgenden Batches ein immer gleicher Zeitabstand (Taktzeit T). Beim zyklischen Ablauf startet in jedem Zyklus (also jedem Zeitintervall der Länge T) immer genau ein Batch, entsprechend wird in jedem Zyklus auch genau ein Batch abgeschlossen. Allerdings übersteigt die Batch-Bearbeitungszeit üblicherweise die Taktzeit. Dementsprechend befinden sich zu jedem Zeitpunkt mehrere Batches gleichzeitig im System.

Bild 1 zeigt einen Ausschnitt eines solchen zyklischen Ablaufs als Gantt Chart. Die Aktivitäten eines Batches sind durch übereinstimmende Färbung gekennzeichnet. Ein Batch umfasst 6 Aktivitäten in 2 Zeitabschnitten (zwischen den beiden Aktivitäten auf der Ressource „Roboter“ befindet sich das Werkstück nicht im hier betrachteten Teil des Systems). Zum Transfer des Werkstücks überschneiden sich die Aktivitäten eines Abschnitts jeweils zeitlich. Ein neuer Batch mit identischem zeitlichen Ablauf startet

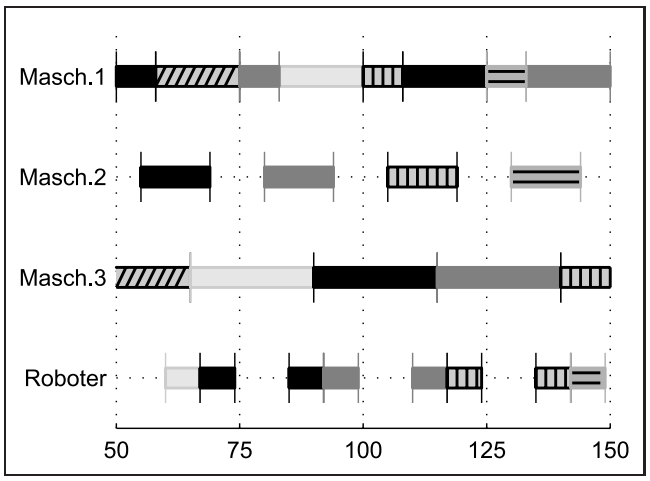

Bild 1: Gantt Chart eines Ausschnitts eines zyklischen Ablaufs mit Taktzeit 25.

alle 25 Zeiteinheiten. Dieser zyklische Ablauf kann in beide Richtungen, d. h. für beliebig viele Batches vor und nach dem abgebildeten Zeitabschnitt fortgesetzt sein.

Im Folgenden wird ein formales Modell des zyklischen Ablaufplanungsproblems eingeführt.

3 Modellbildung

Ein zyklischer Ablauf ist durch

- das für alle Batches identische Zeitschema (*Batch-Zeitschema*) und
- den Zeitabstand zwischen aufeinander folgenden Batches (Taktzeit T)

vollständig definiert. Das Batch-Zeitschema für den einzelnen Batch umfasst n Aktivitäten, denen durch die Abbildung $J : \{1, \ldots, n\} \to \{1, \ldots, m\}$ jeweils eindeutig eine der m Ressourcen zugewiesen ist. $J_i := J(i)$ bezeichnet also die Ressource für Aktivität i.

Im Allgemeinen wird das Batch-Zeitschema durch eine Menge von Ereignissen $x \in \mathcal{X}$ sowie die zugehörigen Ereigniszeitpunkte $t\{x\}$ definiert. Ereignisse x können beispielsweise die Belegung und Freigabe von Ressourcen, der Start bestimmter Operationen innerhalb einer Aktivität oder die Synchronisation mit einer benachbarten Ressource zur Übergabe des Werkstücks sein. Für die Formulierung des Ablaufplanungsproblems sind insbesondere diejenigen Zeitpunkte relevant, die die Ressourcenbelegung und -freigabe für jede Aktivität repräsentieren:

$t\{o_i\} \in \mathbb{R}$ Zeitpunkt der Ressourcenbelegung durch Aktivität i,

$t\{r_i\} \in \mathbb{R}$ Zeitpunkt der Ressourcenfreigabe durch Aktivität i,

wobei

$$t\{r_i\} > t\{o_i\}\,, \qquad o_i, r_i \in \mathcal{X},\ \ i = 1 \ldots n\,. \tag{1}$$

Gemeinsam mit der Taktzeit $T \in \mathbb{R}^+$ definiert das Batch-Zeitschema einen beliebig viele Batches umfassenden zeitlichen Ablauf vollständig: Der Zeitpunkt für ein Ereignis $x^{(\rho)}$ in Batch ρ ergibt sich aus dem Zeitpunkt des

korrespondierenden Ereignisses x im generischen Batch-Zeitschema zu:

$$t\{x^{(\rho)}\} = t\{x\} + \rho T\,, \quad \rho \in \mathbb{Z}\,, \;\; x \in \mathcal{X}. \tag{2}$$

Insbesondere sind die Zeitpunkte für Ressourcenbelegung und -freigabe:

$$\begin{aligned} t\{o_i^{(\rho)}\} &= t\{o_i\} + \rho T \\ t\{r_i^{(\rho)}\} &= t\{r_i\} + \rho T\,, \quad \rho \in \mathbb{Z}\,, \;\; i = 1 \ldots n\,. \end{aligned} \tag{3}$$

Die Aufgabendefinition beinhaltet die für einen Batch erforderlichen Aktivitäten und die jeweils belegten Ressourcen J_i. Wenn auch die Zeitpunkte $t\{x\}$ aller Batchereignisse x und damit $t\{o_i\}$ und $t\{r_i\}$, $i = 1 \ldots n$ festgelegt wären, bliebe als Freiheitsgrad für die Ablaufplanung lediglich die Taktzeit T. Im Allgemeinen wird jedoch nicht das gesamte Batch-Zeitschema, sondern lediglich eine Anzahl (linearer) Abhängigkeiten zwischen den Zeitpunkten $t\{x\}$ der Form

$$t\{x1\} \geq t\{x2\} + \Delta_{x1,x2}, \qquad x1, x2 \in \mathcal{X},\; \Delta_{x1,x2} \in \mathbb{R} \tag{4}$$

vorgegeben sein. Beispiele sind:

$t\{r_i\} \geq t\{o_i\} + d_i$	Mindestdauer einer Aktivität
$t\{x1\} = t\{x2\}$	Synchronisation zwischen zwei Aktivitäten für Werkstück-Transfer
$t\{o_{i1}\} \geq t\{o_{i2}\}$	Reihenfolgevorschrift für Aktivitäten
$t\{o_i\} \geq t\{r_i\} - h_i$	Höchstdauer einer Aktivität (obere Zeitschranke)

Weiterhin kommt die für Aufgaben der Ablaufplanung zentrale Forderung hinzu, dass jede Ressource nur von einer Aktivität gleichzeitig belegt sein kann. Für die effiziente Lösung von großen Ablaufplanungsproblemen ist es entscheidend, für diese Ausschlussbedingungen (disjunctive constraints) eine kompakte Form zu finden. Dies wird im Folgenden erreicht.

Die Ausschlussbedingungen sind genau dann erfüllt, wenn

$$t\{o_{i1}^{(\rho 2)}\} \geq t\{r_{i2}^{(\rho 1)}\} \quad \text{XOR} \quad t\{o_{i2}^{(\rho 1)}\} \geq t\{r_{i1}^{(\rho 2)}\} \tag{5}$$

für jedes (geordnete) Paar von Aktivitäten gilt, die dieselbe Ressource nutzen, d. h. für alle $i1, i2 \in \{1 \ldots n\}$, $\rho 1$, $\rho 2 \in \mathbb{Z}$, $i1 < i2$, $J_{i1} = J_{i2}$. Wegen der Symmetrie in (5) garantiert diese Bedingung auch den Fall $i1 > i2$. Der Sonderfall $i1 = i2$, $\rho 1 \neq \rho 2$ kann durch die Forderung

$$t\{r_i\} - t\{o_i\} \leq T \tag{6}$$

garantiert werden.

Man beachte, dass Bedingung (5) für alle Batches $\rho 1$, $\rho 2 \in \mathbb{Z}$ erfüllt sein muss und sich deshalb in dieser Form nicht für eine algorithmische Umsetzung eignet. Notwendig und hinreichend für Bedingung (5) ist (vgl. [11]):

$$\forall (i1, i2),\; i1 < i2,\; J_{i1} = J_{i2} \;\; \exists\, z_{(i1,i2)} \in \mathbb{Z}\,,$$

$$\text{sodass} \qquad z_{(i1,i2)} T - \big(t\{o_{i2}\} - t\{r_{i1}\}\big) \leq 0 \tag{7}$$

$$(z_{(i1,i2)} + 1) T - \big(t\{r_{i2}\} - t\{o_{i1}\}\big) \geq 0\,. \tag{8}$$

Für jedes (geordnete) Paar von Aktivitäten des generischen Batchschemas wird also eine zusätzliche ganzzahlige Variable $z_{(i1,i2)}$ eingeführt. Hierdurch wird die unendliche Anzahl von Bedingungen (5) durch eine endliche Anzahl von Bedingungen der Form (7), (8) ersetzt. Dieser Schritt ist entscheidend für die Formulierung des Ablaufplanungsproblems als mathematisches Optimierungsproblem.

4 Modellreduktion

Die Darstellung der zeitlichen Freiheitsgrade des Batch-Zeitschemas über die Ereignismenge $\mathcal{X}$ und die Nebenbedingungen (4) lässt sich leicht automatisieren, ist aber häufig stark redundant. Das resultierende Modell kann daher meist durch Hinzufügen weiterer Bedingungen (die die global optimale Taktzeit natürlich nicht ausschließen dürfen) und durch eine geeignete Parametrierung der Freiheitsgrade signifikant vereinfacht werden:

1. *Reduktion der Freiheitsgrade:* Man betrachte zunächst Ereignisse $x1$, die von genau einem Vorgängerereignis $x2$ abhängen. Wie in [11] gezeigt, ändert sich der global optimale Zeittakt nicht, wenn die Ungleichung (4) durch die Gleichung

   $$t\{x1\} = t\{x2\} + \Delta_{x1,x2} \tag{9}$$

   ersetzt wird, $x1$ also schnellstmöglich auf $x2$ folgt.

   Äquivalente Gleichungs-Bedingungen lassen sich für Gruppen von Ereignissen aufstellen, für deren zeitliche Beziehungen wegen des Zusammenwirkens mehrerer Zeilen der Form (4) keine Freiheitsgrade verbleiben. Diese und weitere Modellreduktionsschritte können systematisch anhand einer Digraph-Darstellung durchgeführt werden. Die Knoten der Graphendarstellung repräsentieren dabei die Ereignisse $x \in \mathcal{X}$, die (gerichteten und gewichteten) Kanten die Bedingungen der Form (4). Eine vollständige Darstellung dieser Vorgehensweise findet sich in [11].

2. *Reparametrierung der Freiheitsgrade:* Die Bedingungen der Form (9) beschreiben eine Hyperebene in $\mathbb{R}^{|\mathcal{X}|}$, in der der Vektor $\boldsymbol{t} = [t\{x1\}, t\{x2\}, \ldots]'$ aller Batch-Ereigniszeitpunkte liegen muss. Meist ist die Dimension K dieser Hyperebene deutlich kleiner als $|\mathcal{X}|$. Der Vektor $\boldsymbol{t}$ läßt sich dann mit $K \ll |\mathcal{X}|$ reellwertigen Variablen θ_k, $k = 1, \ldots, K$, parametrieren:

   $$\boldsymbol{t} = \boldsymbol{\xi}_0 + \sum_{k=1}^{K} \boldsymbol{\xi}_k \theta_k\,. \tag{10}$$

   Jeder Ereigniszeitpunkt $t\{x\}$ wird damit als affine Funktion der Variablen θ_k, $k = 1, \ldots, K$, beschrieben:

   $$t\{x\} = \xi_0\{x\} + \sum_{k=1}^{K} \xi_k\{x\} \theta_k\,, \quad x \in \mathcal{X}. \tag{11}$$

Auf der durch (10) bzw. (11) definierten Hyperebene gelten natürlich weiterhin die Ungleichheitsbedingungen der

Form (4). Der Vektor $\boldsymbol{\xi}_0$ kann in der Regel so gewählt werden, dass diese Bedingungen durch

$$\theta_k \geq 0, \quad k = 1 \ldots K \tag{12}$$

nicht verletzt werden. Eine geeignete Wahl der Vektoren $\boldsymbol{\xi}_k$ ermöglicht es weiterhin, die (nicht-negativen) θ_k als eingefügte Wartezeiten im Batch-Zeitschema zu interpretieren.

Setzt man (11) und (12) in (4) ein und lässt redundante Bedingungen weg, ergeben sich nach Einführung entsprechender Abkürzungen die folgenden Ungleichungen in den neuen Variablen θ_k:

$$\vartheta_p - \sum_{k=1}^{K} \kappa_{p,k}\theta_k \geq 0\,, \quad p = 1 \ldots P\,. \tag{13}$$

Um auch Bedingung (6) und die Ausschlussbedingungen (7), (8) in kompakter Form zu schreiben, wird im Folgenden die Menge aller Paare $(i1, i2)$ mit $i1 < i2$ und $J_{i1} = J_{i2}$ auf die Indexmenge $\iota \in \{1 \ldots \iota_{max}\}$ abgebildet.

Mit den Abkürzungen

$$v_{\iota,k} = \xi_k\{o_{i2}\} - \xi_k\{r_{i1}\} \tag{14}$$

$$w_{\iota,k} = \xi_k\{r_{i2}\} - \xi_k\{o_{i1}\} \tag{15}$$

$$v_{\iota,0} = \xi_0\{o_{i2}\} - \xi_0\{r_{i1}\} \tag{16}$$

$$w_{\iota,0} = \xi_0\{r_{i2}\} - \xi_0\{o_{i1}\}\ ,\ \iota = 1 \ldots \iota_{max} \tag{17}$$

und

$$u_{i,k} = \xi_k\{r_i\} - \xi_k\{o_i\} \tag{18}$$

$$u_{i,0} = \xi_0\{r_i\} - \xi_0\{o_i\}\ ,\ i = 1 \ldots n \tag{19}$$

ergibt sich die folgende kompakte Darstellung für (6) und die Ausschlussbedingungen (7) und (8):

$$u_{i,0} + \sum_{k=1}^{K} u_{i,k}\,\theta_k \leq T, \quad i = 1 \ldots n \tag{20}$$

$$z_\iota T - v_{\iota,0} - \sum_{k=1}^{K} v_{\iota,k}\,\theta_k \leq 0, \quad \iota = 1 \ldots \iota_{max} \tag{21}$$

$$(z_\iota + 1)T - w_{\iota,0} - \sum_{k=1}^{K} w_{\iota,k}\,\theta_k \geq 0, \quad \iota = 1 \ldots \iota_{max} \tag{22}$$

5 Optimale Ablaufplanung

Auf Basis des mathematischen Modells (2), (11), (13) und (20)–(22) für den zyklischen Betrieb kann das Ablaufplanungsproblem nun als gemischt-ganzzahliges Optimierungsproblem formuliert werden.

Ziel der Ablaufplanung ist die Maximierung des Durchsatzes bzw. die Minimierung der Taktzeit T. Außer durch die zu minimierende Zielfunktion T wird das Problem durch die Nebenbedingungen (13), (20)–(22) in den Variablen $\theta_k \in \mathbb{R}_0^+$ und $z_\iota \in \mathbb{Z}$ definiert. Die Ungleichungen (13) und (20) repräsentieren Einschränkungen für das Zeitschema eines einzelnen Batches, (21) und (22) die Ausschlussbedingungen, die garantieren, dass jede Ressource im zyklischen Ablauf zu jedem Zeitpunkt von nur einer Aktivität belegt wird.

Damit ergibt sich ein nichtlineares gemischt-ganzzahliges Optimierungsproblem (mixed integer nonlinear program, MINLP) mit den Variablen $z_\iota \in \mathbb{Z},\ \iota = 1 \ldots \iota_{max},\ T \in \mathbb{R}^+$ und $\theta_k \in \mathbb{R}_0^+,\ k = 1 \ldots K$. Dieses nichtlineare Problem lässt sich in einem letzten Schritt über die Transformation

$$\bar{T} := \frac{1}{T}\,,\ \bar{\theta}_k := \frac{\theta_k}{T}\,,\ k = 1 \ldots K \tag{23}$$

in lineare Form überführen. Dabei wird weder die Anzahl der benötigten Variablen noch die Lösungsmenge des Problems verändert. Das so entwickelte lineare gemischt-ganzzahlige Optimierungsproblem (mixed integer linear program, MILP) (24)–(27) kann mit Verfahren der mathematischen Optimierung global optimal gelöst werden und liefert somit einen zyklischen Ablaufplan mit optimalem Durchsatz.

$$\left(\bar{T} \in \mathbb{R}^+,\ \bar{\theta}_k \in \mathbb{R}_0^+,\ k = 1 \ldots K,\ z_\iota \in \mathbb{Z},\ \iota = 1 \ldots \iota_{max}\right)$$

Max $\bar{T}$ s. t.

$$z_\iota - v_{\iota,0}\bar{T} - \sum_{k=1}^{K} v_{\iota,k}\bar{\theta}_k \leq 0 \tag{24}$$

$$z_\iota + 1 - w_{\iota,0}\bar{T} - \sum_{k=1}^{K} w_{\iota,k}\bar{\theta}_k \geq 0, \quad \iota = 1 \ldots \iota_{max} \tag{25}$$

$$\vartheta_p\bar{T} - \sum_{k=1}^{K} \kappa_{p,k}\bar{\theta}_k \geq 0, \quad p = 1 \ldots P \tag{26}$$

$$1 - u_{i,0}\bar{T} - \sum_{k=1}^{K} u_{i,k}\bar{\theta}_k \geq 0, \quad i = 1 \ldots n \tag{27}$$

Die Darstellung (24)–(27) erlaubt die Berechnung global optimaler Lösungen für große Aufgabenstellungen, beispielsweise aus dem Bereich des High-Throughput-Screening, in wenigen Sekunden. Die Formulierung kann noch weiter geschärft werden, indem das Optimierungsproblem um Schranken für die Variablen $\bar{T}$ und z_ι und Schnittebenen (cuts) ergänzt wird [11].

Tabelle 2 zeigt Problemgröße und Rechenzeit für das Beispiel aus Bild 1 und drei reale Anwendungsbeispiele zunehmenden Umfangs aus dem Bereich des High-Throughput-Screening. Die Beispiele werden durch die Anzahl der Ressourcen m, die Anzahl der Aktivitäten pro Batch n und die Anzahl der Aktivitätenpaare ι_{max} charakterisiert; die Größe des letztlich zu lösenden Optimierungsproblems ist durch die Anzahl reellwertiger Variablen (#real), die Anzahl ganzzahliger Variaben (#int) und die Anzahl der Nebenbedingungen (#con) gekennzeichnet. Die gemischt-ganzzahligen Optimierungsprobleme wurden mit dem Solver GAMS/CPLEX [6] auf einem 2 GHz-Linux-PC mit 1 GByte Arbeitsspeicher gelöst. Das sehr große Beispiel

Tabelle 2: Größe und Rechenzeit für Beispiele.

Beispiel	m	n	ι_{max}	#real	#int	#con	t_{solve}
Bild 1	4	6	2	4	2	15	< 1 s
HTS	5	30	130	28	85	1165	< 1 s
HTS groß	18	57	143	52	131	1051	< 1 s
HTS XXL	18	87	350	78	321	2402	122 s

‚HTS XXL' beschreibt eine Scheduling-Aufgabenstellung, bei der auf einer Anlage mehrere Screening-Aufgaben bei optimalem Durchsatz parallel ausgeführt werden sollen. Auch hier ist eine Berechnung des globalen Optimums, also des bestmöglichen zyklischen Ablaufplans, noch in akzeptabler Rechenzeit möglich, jedoch steigt die Komplexität und damit die Rechenzeit mit zunehmender Größe stark an. Weitere Beispiele aus der pharmazeutischen Industrie werden in [11] untersucht. Für alle Beispiele konnte die global optimale Lösung in wenigen Sekunden berechnet werden. Gegenüber den vorher bekannten von Hand ermittelten Ablaufplänen konnte der Durchsatz um bis zu 169% gesteigert werden.

6 Pooling-Ressourcen

In den vorangegangenen Abschnitten haben wir uns auf die Betrachtung von Ressourcen beschränkt, die zu jedem Zeitpunkt von maximal einer Aktivität belegt sein können. Die Modellbildung wird nun durch eine Klasse von Ressourcen erweitert, die sich dadurch auszeichnen, dass sie in einem Arbeitsgang eine festgelegte Anzahl $C > 1$ von Werkstücken gleichzeitig verarbeiten *müssen*. Wir bezeichnen solche Ressourcen als *Pooling-Ressourcen* und die Anzahl C der gleichzeitig zu bearbeitenden Werkstücke als *Poolingkapazität*. Typische Beispiele solcher Pooling-Ressourcen sind Öfen, die aus ökonomischen Gründen voll ausgelastet werden müssen, oder Zentrifugen mit Aufnahmeplätzen für mehrere Proben, die aus technischen Gründen (Vermeidung von Unwucht) stets voll bestückt arbeiten müssen.

Wir unterscheiden zwei Arten von Pooling-Ressourcen [8]: Bei Pooling-Ressourcen vom *Typ 1* müssen sämtliche Werkstücke entladen sein, bevor das erste Werkstück des darauf folgenden Arbeitsgangs beladen werden kann. Beim *Typ 2* können frei gewordene Plätze sofort wieder belegt werden, auch wenn noch nicht alle Werkstücke des vorherigen Arbeitsgangs entfernt wurden. Man beachte, dass die Werkstücke aufgrund der Natur des Poolings nicht mehr zyklisch bearbeitet werden. Im Vergleich zu *Typ 1* besitzen Pooling-Ressourcen vom *Typ 2* zusätzliche Freiheitsgrade. Bei Verwendung von *Typ 2* Ressourcen ist deswegen unter Umständen eine kleinere Zykluszeit möglich.

Pooling-Ressourcen des Typs 1 lassen sich in das Optimierungsproblem (24)–(27) integrieren, indem mehrere Batches zu einem *High-level Batch* zusammengefasst werden. Die so zusammengefassten einzelnen Batches bezeichnen wir als *Low-level Batches*.[3] Interpretiert man den resultierenden High-level Batch als Batch im Sinne von Abschnitt 2 und 3, so kann mit identischem Verfahren ein MILP der Form (24)–(27) erstellt werden. Die Pooling-Ressource ist dann eine einfache Ressource im Sinne der Definition in Abschnitt 2, und es existieren Präzedenzbeziehungen zwischen Ereignissen der Pooling-Aktivität und Ereignissen aller beteiligten Low-level Batches.

Beispiel: Man betrachte die Probleminstanz aus dem Beispiel in Bild 1, die um eine Pooling-Ressource „Ofen" des Typs 1 mit Kapazität $C = 2$ erweitert ist. Nach den Arbeitsschritten auf Maschine 1 und 2 werden die Werkstücke mit dem Roboter auf die Pooling-Ressource transportiert. Dem Arbeitsgang auf der Pooling-Ressource schließt sich ein weiterer Arbeitsschritt auf Maschine 3 an, wobei wieder der Roboter für den Transport benötigt wird. Die optimale Lösung des Schedulingproblems ist als Gantt Chart in Bild 2 dargestellt. Jeder High-level Batch (je in einer Graustufe gekennzeichnet) besteht aus zwei Low-level Batches. Es ist zu erkennen, dass die Pooling-Ressource „Ofen" vom Beginn des Beladens des ersten Werkstücks bis zum Ende des Entladens des zweiten Werkstücks belegt ist.

Bei Pooling-Ressourcen des Typs 2 müssen die einzelnen Plätze jeweils als eigene Ressourcen mit eigenen Aktivitäten modelliert werden. Ein High-level Batch besteht dann aus mehreren Low-level Batches, in denen die Pooling-Aktivität durch eine Slot-Aktivität ersetzt wird, sowie einer Pooling-Aktivität, die lediglich zu den Slot-Aktivitäten der Low-level Batches Präzedenzbeziehungen unterhält. Wie in Bild 3 gezeigt, kann die Pooling-Aktivität „Heizen" auf der Ressource „Ofen" erst nach Beginn aller Slot-Aktivitäten beginnen und muss vor Ende aller Slot-Aktivitäten beendet sein. Durch zusätzliche Bedingungen im Ereignismodell muss also sichergestellt werden, dass die Zeitintervalle der Slot-Aktivitäten das Zeitintervall der auf der

[3] Die Anzahl der zusammenzufassenden Low-level Batches ist sinnvollerweise gleich dem kleinsten gemeinsamen Vielfachen aller im Prozess auftretenden Poolingkapazitäten zu wählen. Im Folgenden gehen wir der Einfachheit halber allerdings davon aus, dass der Prozess nur eine Pooling-Ressource enthält.

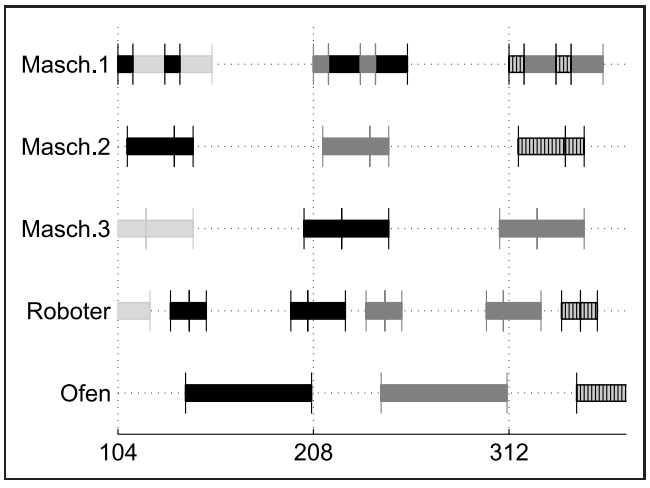

Bild 2: Zyklischer Ablauf mit Pooling-Ressource des Typs 1.

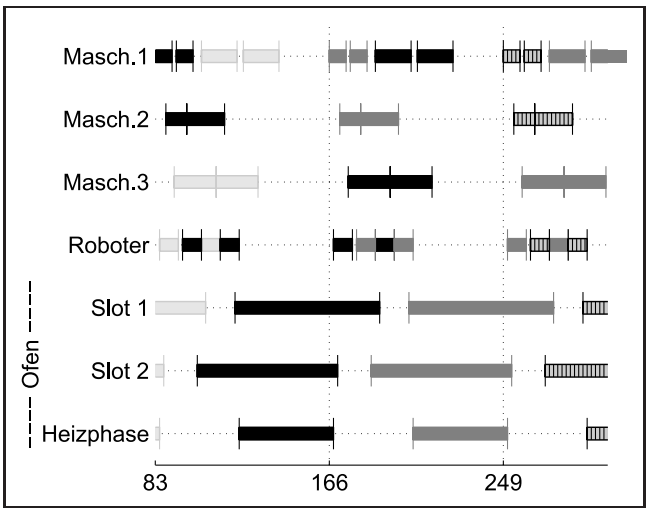

Bild 3: Zyklischer Ablauf mit Pooling-Ressource des Typs 2.

Pooling-Ressource ausgeführten Aktivität enthalten: Sei $\mathcal{P}$ die Menge aller Indizes der Slot-Aktivitäten der Pooling-Ressource mit Kardinalität $|\mathcal{P}| = C$. Dann gelten für den Start- und Endzeitpunkt $(t\{o_{pool}\}, t\{r_{pool}\})$ der Poolingaktivität folgende Beziehungen:

$$t\{o_{pool}\} \geq t\{o_i\}, \quad \forall i \in \mathcal{P}$$

$$t\{r_{pool}\} \leq t\{r_i\}, \quad \forall i \in \mathcal{P}$$

$$t\{r_{pool}\} \geq t\{o_{pool}\} + d_{pool}.$$

Bild 3 zeigt das Gantt Chart unseres Beispiels, wenn die Pooling-Ressource vom Typ 2 ist. Die beiden Plätze des Ofens sind als eigene Ressource modelliert, die vom Beginn der Beladung bis zum Ende der Entladung belegt sind. Die eigentliche Pooling-Aktivität auf dem Ofen kann beginnen, sobald der Transport zum letzten Platz abgeschlossen ist. Im Vergleich zu Bild 2, in dem der Ofen als Pooling-Ressource des Typs 1 betrachtet wurde, konnte die Zykluszeit T von 104 auf 83 Zeiteinheiten reduziert werden. Die intuitive Erklärung für diese Reduktion ist, dass die Ressource „Ofen/Slot 2" bereits wieder belegt werden kann, obwohl sich noch ein Werkstück in der Ressource „Ofen/Slot 1" befindet.

Die Zusammenfassung der Low-level Batches zu einem High-level Batch bewirkt allerdings, dass die Anzahl der reellen Entscheidungsvariablen θ_k ungefähr C-fach so groß ist wie im Fall ohne Pooling. Dies führt zu einem deutlichen Anstieg des Rechenaufwandes und kann deshalb selbst bei Problemen moderater Größe inakzeptable Laufzeiten der Optimierungs-Software nach sich ziehen.

Der notwendige Rechenaufwand kann erheblich reduziert werden, wenn zusätzliche zeitliche Bedingungen zwischen den Low-Level Batches eine *innere Zyklizität* erzwingen. Sei T^* die *innere Zykluszeit*. Dann fordern wir, dass gleiche Ereignisse unterschiedlicher Low-level Batches desselben High-level Batch mit Zeitversatz T^* erfolgen:

$$t\left\{x_i^{(c)}\right\} = t\left\{x_i^{(0)}\right\} + cT^* \quad \forall\, c \in \{0, \ldots, C-1\},$$

wobei $x_i^{(c)}$ und $x_i^{(0)}$ gleiche Ereignisse in den Low-level Batches mit Index (c) und (0) desselben High-level Batch sind.

Hierdurch wird die diskutierte Vervielfachung der reellen Entscheidungsvariablen θ_k vermieden. Im Vergleich zu einem Problem ohne Pooling erhält man nur eine zusätzliche reelle Entscheidungsvariable, nämlich die innere Zykluszeit T^*. Im Vergleich zum Pooling ohne innere Zyklizität reduziert sich der notwendige Aufwand zur Berechnung der zeitoptimalen Lösung deswegen drastisch. In vielen Fällen hat die zusätzliche Forderung der inneren Zyklizität nur wenig Einfluss auf die sich ergebende Taktzeit T und ist oft aus prozesstechnischen Erwägungen sogar erwünscht. Für unser Beispiel mit Pooling-Ressource vom *Typ 2* und Kapazität $C = 6$ ergibt sich eine Zykluszeit von $T = 183$ Zeiteinheiten bei einer Rechenzeit von 8320 sec. Erzwingt man hingegen innere Zyklizität, so verringert sich die Rechenzeit auf 53 sec. Die resultierende Zykluszeit ist mit $T = 188$ Zeiteinheiten nur unwesentlich größer als im global optimalen Fall.

7 Zusammenfassung

In diesem Beitrag wurde ein Ansatz zur Lösung zeitoptimaler zyklischer Ablaufplanungsprobleme beschrieben. Die Vorgehensweise wurde zunächst für Ressourcen der Kapazität 1 erläutert. Zentraler Schritt bei der Modellbildung war eine geeignete Formulierung der Ausschlussbedingungen. Diese stellen sicher, dass zu jedem Zeitpunkt höchstens eine Aktivität (aus beliebig vielen Batches) auf jeder Ressource erfolgt. Das Modell wird durch Reduktion der Freiheitsgrade und Reparametrierung so vereinfacht, dass zumindest eine (global) optimale Lösung erhalten bleibt. Das resultierende Problem lässt sich als gemischt-ganzzahliges nichtlineares Optimierungsproblem (MINLP) und – nach geeigneter Transformation – als gemischt-ganzzahliges *lineares* Problem (MILP) schreiben. Letzteres kann dann mit bekannten Verfahren und Werkzeugen effizient gelöst werden.

Für die Behandlung paralleler Ressourcen (multi-capacity resources) wird auf [12] verwiesen. Solche Ressourcen können (müssen aber nicht) durch mehrere Werkstücke gleichzeitig belegt werden.

Im vorliegenden Beitrag wurde gezeigt, wie sich die in vielen Anwendungen wichtigen Pooling-Ressourcen in das beschriebene Verfahren integrieren lassen. Solche Ressourcen *müssen* bei jeder Aktivierung eine vorgegebene Zahl von Werkstücken aufnehmen und verarbeiten.

Schließlich sei darauf hingewiesen, dass in diesem Beitrag Ablaufplanungsprobleme auf einem durch zeitbehaftete diskrete Ereignisse vollständig charakterisierten Abstraktionsniveau gelöst wurden. Lässt sich die kontinuierliche Dynamik einzelner Aktivitäten nicht unabhängig von den ereignisdiskreten Aspekten untersuchen, erhält man ein Optimierungsproblem für ein hybrides dynamisches System.

Literatur

[1] A. Alle, L. G. Papageorgiou, and J. M. Pinto. A mathematical programming approach for cyclic production and cleaning scheduling of multistage continuous plants. *Computers & Chemical Engineering, UK*, 28(1–2):3–15, 2004.

[2] P. Brucker. *Scheduling Algorithms*. Springer, 2007.

[3] H. Chen, C. Chu, and J.-M. Proth. Cyclic scheduling of a hoist with time window constraints. *IEEE Transactions on Robotics & Automation*, 14(1):144–152, 1998.

[4] Y. Crama, V. Kats, J. van de Klundert, and E. Levner. Cyclic scheduling in robotic flowshops. *Annals of Operations Research*, 96:97–124, 2000.

[5] A. Di Febbraro, R. Minciardi, and S. Sacone. Deterministic timed event graphs for performance optimization of cyclic manufacturing processes. *IEEE Transactions on Robotics & Automation*, 13(2):169–181, 1997.

[6] GAMS/CPLEX, 2008. http://www.gams.com.

[7] C. Hanen and A. Munier. Cyclic scheduling on parallel processors: An overview. In P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu, editors, *Scheduling Theory and its applications*, pages 193–226. Wiley, New York, 1995.

[8] C. Horst. Throughput-optimal cyclic scheduling with pooling resources. Diplomarbeit, Institut für Automatisierungstechnik, Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2006.

[9] W. T. Huh, G. Janakiraman, P. L. Jackson, and N. Sawhney. Minimizing flow time in cyclic schedules for identical jobs with acyclic precedence: the bottleneck lower bound. *Operations Research Letters*, 31(5):366–374, 2003.

[10] J. Kallrath. *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis*. Vieweg, 2002.

[11] E. Mayer. *Globally Optimal Schedules for Cyclic Systems with Non-Blocking Specification and Time Window Constraints*. Shaker, 2007.

[12] E. Mayer and J. Raisch. Throughput-optimal scheduling for cyclically repeated processes. In *MMAR2003 – 9th IEEE International Conference on Methods and Models in Automation and Robotics*, pages 871–876, Miedzyzdroje, Poland, August 2003.

[13] E. Mayer and J. Raisch. Time-optimal scheduling for high throughput screening processes using cyclic discrete event models. *MATCOM – Mathematics and Computers in Simulation*, 66(2–3):181–191, 2004.

[14] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Prentice Hall, 2002.

[15] C. N. Potts and M. Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228–249, 2000.

[16] R. Roundy. Cyclic schedules for job shops with identical jobs. *Mathematics of Operations Research*, 17(4):842–65, 1992.

[17] J.-W. Seo and T.-E. Lee. Steady-state analysis and scheduling of cyclic job shops with overtaking. *International Journal of Flexible Manufacturing Systems*, 14(4):291–318, 2002.

Manuskripteingang: 12. November 2007.

Dr.-Ing. Eckart Mayer war zum Zeitpunkt der Entstehung dieser Arbeit wissenschaftlicher Mitarbeiter am Max-Planck-Institut für Dynamik komplexer technischer Systeme in Magdeburg. Nach seiner Promotion am Fachgebiet Regelungssysteme der Technischen Universität Berlin ist er jetzt für die Robert Bosch GmbH, Stuttgart, tätig.

Adresse: Technische Universität Berlin, Fachgebiet Regelungssysteme, EN 11, Einsteinufer 17, 10587 Berlin,
E-Mail: mayer@control.tu-berlin.de

Dr. Kai Wulff ist wissenschaftlicher Mitarbeiter am Fachgebiet Regelungssysteme an der Fakultät für Elektrotechnik und Informatik der TU Berlin. Hauptarbeitsgebiete: Stabilitätstheorie geschalteter Systeme.

Adresse: Technische Universität Berlin, Fachgebiet Regelungssysteme, EN 11, Einsteinufer 17, 10587 Berlin, Tel.: +49-(0)30 314 22276,
E-Mail: wulff@control.tu-berlin.de

Dipl.-Ing. Christoph Horst war zum Zeitpunkt der Entstehung dieser Arbeit Diplomand am Fachgebiet Systemtheorie technischer Prozesse der Otto-von-Guericke-Universität Magdeburg und wissenschaftlicher Mitarbeiter des Fachgebiets Regelungssysteme der Technischen Universität Berlin. Er ist jetzt für die dSpace GmbH, Paderborn, tätig.

Adresse: dSpace GmbH, Technologiepark 25, 33100 Paderborn,
E-Mail: chorst@dspace.de

Prof. Dr.-Ing. Jörg Raisch leitet das Fachgebiet Regelungssysteme an der Fakultät für Elektrotechnik und Informatik der TU Berlin. Er ist externes wissenschaftliches Mitglied des Max-Planck-Instituts für Dynamik komplexer technischer Systeme in Magdeburg und leitet dort die Fachgruppe System- und Regelungstheorie.

Adresse: Technische Universität Berlin, Fachgebiet Regelungssysteme, EN 11, Einsteinufer 17, 10587 Berlin, Tel.: +49-(0)30 314 22945, Fax: +49-(0)30 314 21137,
E-Mail: raisch@control.tu-berlin.de, URL: www.control.tu-berlin.de