

HUMBOLDT-UNIVERSITÄT ZU BERLIN



Institut für Informatik

# Optimierung der Sweep-Line-Methode

Studienarbeit

Robert Prüfer

Betreuer: Daniela Weinberg, Christian Stahl

Berlin, den 13. März 2009



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
<b>2</b>	<b>Grundlagen</b>	<b>7</b>
2.1	Definitionen und Terminologie . . . . .	7
2.2	Die Sweep-Line-Methode . . . . .	8
2.3	Algebraische Berechnung von Progress-Werten . . . . .	10
2.3.1	Beispiel . . . . .	11
2.3.2	Geometrische Interpretation . . . . .	11
2.3.3	Ansätze zur Optimierung . . . . .	12
<b>3</b>	<b>Geometrisch basierter Ansatz zur Berechnung der Offset-Werte</b>	<b>15</b>
3.1	Beispiel zur Veranschaulichung . . . . .	17
3.2	Probleme mit Nullen als Offset-Werte . . . . .	19
<b>4</b>	<b>Optimierung der Anzahl negativer Offset-Werte</b>	<b>21</b>
4.1	Das Maximum Feasible Subsystem Problem . . . . .	22
4.1.1	Algorithmische Lösungsmethoden . . . . .	23
4.1.2	Schwierigkeiten bei der Anwendung von MAXFS bezüglich der Optimierung . . . . .	24
4.2	Beispiel: Petrinetz ohne Regresstransitionen . . . . .	25
4.3	Beispiel: Petrinetz mit Regresstransitionen und die geometrische Interpretation von MAXFS . . . . .	26
<b>5</b>	<b>Schlussbemerkungen</b>	<b>29</b>
5.1	Zusammenfassung . . . . .	29
5.2	Ausblick . . . . .	29



# 1 Einleitung

Für die Spezifikation eines Systems werden heutzutage oft formale Methoden, denen mathematische Strukturen zugrunde liegen, genutzt. Aufgrund ihrer mathematischen Fundierung sind sie oftmals gut für die automatische Analyse von entsprechend spezifizierten Systemen geeignet. Für viele dieser formalen Methoden kann ein Zustandsraum, also die Menge aller Zustände, die in einem System möglich sind, erstellt werden. Allerdings stößt man hierbei schnell auf das Problem der sogenannten *Zustandsraumexplosion* [Val90]: Ein Zustandsraum kann schon für ein relativ kleines System eine große Anzahl an Zuständen enthalten, so dass die verfügbaren Speicherkapazitäten der verwendeten Rechner überschritten werden und es nicht möglich ist, den kompletten Zustandsraum im Speicher zu halten bzw. überhaupt zu berechnen. Zur Überprüfung gewisser Eigenschaften des modellierten Systems ist es oft jedoch nötig zu wissen, ob ein bestimmter Zustand dieses Systems erreichbar ist (sich also im Zustandsraum befindet) oder nicht.

Um den Speicherplatzverbrauch bei der Suche nach erreichbaren Zuständen zu reduzieren, wurden viele verschiedene Techniken entwickelt. Die Sweep-Line-Methode ist eine solche Technik; sie wird in dieser Arbeit betrachtet. Die Idee dieses Ansatzes ist es, Zustände, die einmal gefunden und für die weitere Berechnung des Zustandsraumes nicht mehr benötigt werden, zu löschen. Um zu entscheiden, welche Zustände gelöscht werden können, wird eine *Progress Measure* benötigt, die im gewissen Sinne den „Fortschritt“ der Durchmusterung des Zustandsraumes wiedergibt. Beispielsweise werden in dem in Abb. 1.1 dargestellten Zustandsraum alle schwarz markierten Zustände nicht mehr zur weiteren Durchmusterung des Zustandsraumes benötigt; weil der ihnen zugewiesene Progress-Wert eine gewisse Grenze unterschreitet, können sie gelöscht werden. In der Originalpublikation zur Sweep-Line-Methode [Mai03] wird auf eine automatische Bestimmung der Progress Measure nicht eingegangen und zunächst vorgeschlagen, diese mit Bezug auf das jeweilige System von Hand zu erstellen.

Ein Verfahren zur automatischen Berechnung von Progress-Werten für Petrinetze (eine weit ver-

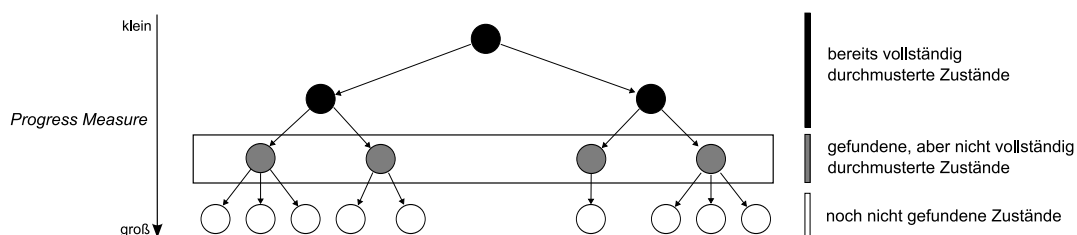


Abbildung 1.1: Eine Momentaufnahme der Zustandsraumdurchmusterung durch Breitensuche mit der Sweep-Line-Methode

## 1 Einleitung

breitete formale Methode) wurde bereits entwickelt [Sch04]. Diese Berechnung lässt allerdings verschiedene Freiheitsgrade zu, so dass sie für viele Petrinetz-Modelle nicht die optimale Progress Measure berechnet. Durch eine verbesserte Berechnung wäre es also möglich, bei der Suche nach erreichbaren Zuständen von Petrinetzen mehr Speicherplatz einzusparen und somit die Chance zu erhöhen, dass auch bei Netzen mit großen Erreichbarkeitsgraphen die Erreichbarkeit eines Zustands entschieden werden kann.

In der vorliegenden Arbeit soll ein Ansatz für ein Kriterium zur Optimierung dieser Berechnung, die Minimierung der Anzahl negativer Offset-Werte, vorgestellt werden. Ziel dieser Minimierung ist es, zu verhindern, dass Teile des Zustandsraumes unnötig oft durchlaufen werden. Nach einigen notwendigen Definitionen wird zunächst die Sweep-Line-Methode und im Anschluss die bereits existierende algebraische Methode zur Berechnung von Progress-Werten für Petrinetze erläutert. Nachfolgend wird ein neuer, geometrisch basierter Ansatz zur Berechnung von Offset-Werten beschrieben, auf dessen Grundlage dann die Minimierung der Anzahl negativer Offset-Werte betrachtet wird.

## 2 Grundlagen

### 2.1 Definitionen und Terminologie

An dieser Stelle werden wir einige notwendige Begriffe und Definitionen einführen.

Im Folgenden gilt für die Menge  $\mathbb{N}$  der natürlichen Zahlen:  $0 \in \mathbb{N}$ .

Ein *Petrinetz* beschreiben wir als 5-Tupel  $(P, T, F, W, s_0)$ .  $P = (P_0, \dots, P_n)$  und  $T = (t_0, \dots, t_k)$  sind endliche Mengen, wobei  $P$  die Plätze und  $T$  die Transitionen des Petrinetzes enthält (die Tiefstellung der Indizes entfällt bei den Abbildungen). Die Relation  $F \subseteq (P \times T) \cup (T \times P)$  repräsentiert die Kanten zwischen Plätzen und Transitionen; für  $(x, y) \notin F$  setzen wir  $W(x, y) = 0$ . Die Funktion  $W : F \rightarrow \mathbb{N} \setminus \{0\}$  ordnet den Kanten Gewichte zu; besitzt eine Kante das Gewicht 1, wird dieses in der grafischen Darstellung nicht angegeben.  $s_0$  repräsentiert die Anfangsmarkierung des Petrinetzes, wobei eine Markierung durch die Funktion  $s : P \rightarrow \mathbb{N}$  beschrieben wird. Zu jedem Petrinetz kann man die *Inzidenzmatrix*  $M = |P| \times |T|$  betrachten; die Einträge der Matrix stehen für die Anzahl der Marken, die auf den jeweiligen Plätzen beim Schalten einer Transition erzeugt bzw. verbraucht werden. Mit  $M(x, y)$  bezeichnen wir den Eintrag der  $x$ -ten Zeile und  $y$ -ten Spalte von  $M$ . Werden beim Schalten der Transition  $t_i$   $m$  Marken, die auf dem Platz  $P_j$  liegen, verbraucht, dann gilt  $M(j+1, i+1) = -m$ . (Für  $i = 0$  und  $j = 0$  entspricht  $M(j+1, i+1)$  also dem Eintrag „oben links“ in der ersten Zeile und ersten Spalte von  $M$ ; „nullte“ Spalten bzw. Zeilen existieren nicht.) Falls an selber Stelle  $m$  Marken erzeugt werden, folgt  $M(j+1, i+1) = m$ . Werden am Platz  $P_j$  beim Schalten der Transition  $t_i$  keine Marken verbraucht oder erzeugt, gilt  $M(j+1, i+1) = 0$ . Der  $i$ -te Transitionsvektor (also Spaltenvektor von  $M$ ) wird als  $\Delta t_i$  bezeichnet.

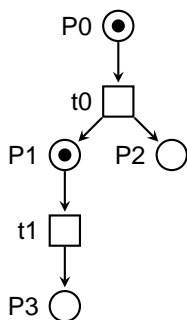


Abbildung 2.1: Ein Petrinetz mit den Plätzen  $P = \{P_0, P_1, P_2, P_3\}$  sowie den Transitionen  $T = \{t_0, t_1\}$

## 2 Grundlagen

Zur Veranschaulichung der Definitionen ist in Abb. 2.1 das Petrinetz mit der Inzidenzmatrix

$$M = \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

sowie der Anfangsmarkierung  $(s_0(P_0), s_0(P_1), s_0(P_2), s_0(P_3)) = (1, 1, 0, 0)$  dargestellt.

In dieser Arbeit werden Petrinetze fast ausschließlich anhand ihrer Inzidenzmatrix betrachtet, da die tatsächliche Markierung eines Netzes für unsere Ausführungen nicht relevant ist.

Zu einem gegebenen Petrinetz kann der *Erreichbarkeitsgraph*  $G = (S, K, s_0)$  erstellt werden, wobei  $S$  die Menge der Zustände und  $K$  die Menge der Kanten dieses Graphen ist. In  $S$  befinden sich genau die erreichbaren Markierungen des Petrinetzes. Die Anfangsmarkierung  $s_0$  ist der Startzustand dieses Graphen. Ist bei einer Markierung  $s$  die Transition  $t_i$  aktiviert, kann durch das Schalten dieser Transition die Markierung  $s' = s + \Delta t_i$  erreicht werden; daher gilt  $(s, s') \in K$ .  $s'$  bezeichnen wir als Folgezustand; alle Zustände, die von  $s$  aus erreichbar sind, bezeichnen wir als transitive Folgezustände.

$r(M)$  sei der *Rang der Matrix*  $M$ ; dieser ist als die größte Menge linear unabhängiger Spaltenvektoren definiert.

### 2.2 Die Sweep-Line-Methode

Die Sweep-Line-Methode ist eine Technik zur Reduktion von Speicherplatzverbrauch bei der Suche nach erreichbaren Zuständen in Transitionssystemen. Für unsere Zwecke genügt es, als Transitionssystem den Erreichbarkeitsgraphen eines Petrinetzes zu betrachten; aus diesem Grund werden wir im Folgenden, mit Ausnahme des Satzes am Ende dieses Abschnitts, nur den Begriff „Erreichbarkeitsgraph“ verwenden.

Es existieren zwei Varianten der Sweep-Line Methode: Die Basisvariante sowie die verallgemeinerte Sweep-Line-Methode. Wir betrachten nun die Basisvariante, bevor wir uns der verallgemeinerten Variante zuwenden, welche die Grundlage unserer weiteren Betrachtungen bilden wird.

Die grundlegende Idee der Sweep-Line-Methode ist, dass in jedem Erreichbarkeitsgraphen eine Art Fortschritt ausfindig gemacht werden kann. Für die Suche nach erreichbaren Zuständen bedeutet dies, dass Zustände, die bereits gefunden wurden, ab einem gewissen Punkt des Fortschritts nicht mehr benötigt werden. Dieser Punkt ist erreicht, wenn auch alle Folgezustände eines Zustands durchmustert wurden. Somit können wir in einem System zwischen drei Klassen von Zuständen unterscheiden: (1) Zustände, die noch nicht gefunden wurden; (2) Zustände, die gefunden wurden und deren Folgezustände bereits alle durchmustert wurden; (3) Zustände, die gefunden wurden und deren Folgezustände noch nicht alle durchmustert wurden. Letztere Klasse wird im Folgenden mit *Front* bezeichnet. Die drei Klassen sind in Abb. 2.2 anhand eines Beispiels dargestellt.

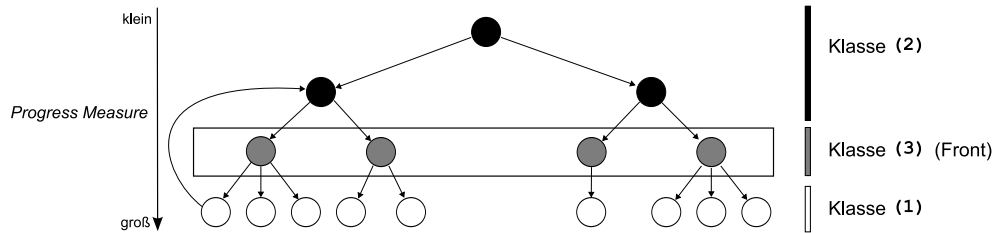


Abbildung 2.2: Ein Erreichbarkeitsgraph, für den die Basisvariante der Sweep-Line-Methode aufgrund der Regresskante nicht terminiert

**Definition 1 (Progress Measure)** Eine *Progress Measure* ist eine Funktion  $p : S \rightarrow A$ , wobei  $S$  die Menge der Zustände eines Erreichbarkeitsgraphen ist und  $A \in \{\mathbb{N}, \mathbb{Q}\}$ .<sup>1</sup>

Die Progress Measure sei zunächst monoton, d.h. für  $(s, s') \in K$  gelte  $p(s) \leq p(s')$ . Hieraus wird ersichtlich, dass sie den „Fortschritt“ bei der Durchmusterung eines Erreichbarkeitsgraphen angibt. Der Progress-Wert der Zustände, die bereits inklusive aller Folgezustände vollständig durchmustert wurden, ist kleiner als der minimale Progress-Wert der Zustände der Front. Aufgrund der Monotonie von  $p$  können jetzt alle Zustände der Klasse (2) gelöscht werden; sie spielen für die weitere Durchmusterung des Erreichbarkeitsgraphen keine Rolle mehr. Hier wird das Prinzip der Sweep-Line-Methode deutlich: Je weiter der Erreichbarkeitsgraph durchmustert wird, desto weiter schreitet auch die Front voran; alle Zustände, die hinter der Front liegen, wurden gelöscht, so dass nur noch Zustände im Speicher gehalten werden, die für die weitere Durchmusterung relevant sind.

Allerdings treten bei der Anwendung dieser Methode Probleme auf, sobald im Erreichbarkeitsgraphen Kanten auftreten, die zu Zuständen der Klasse (2) führen; da diese Zustände bereits aus dem Speicher gelöscht wurden, ist auch die Information, dass sie bereits gefunden wurden, verloren gegangen. Folglich werden diese Zustände sowie alle ihre Folgezustände und transitiven Folgezustände noch einmal durchmustert und ein paar Schritte später wieder gelöscht – die Basisvariante terminiert also für solche Erreichbarkeitsgraphen nicht (siehe Abb. 2.2).

Um dieses Problem zu lösen, wurde die Basisvariante zur verallgemeinerten Sweep-Line-Methode weiterentwickelt. Hierbei wird auf das Monotoniekriterium der Progress Measure verzichtet; es kann prinzipiell jede beliebige Funktion als Progress Measure verwendet werden. Diese Methode kann auch für Erreichbarkeitsgraphen, die Kanten enthalten, welche zu Zuständen der Klasse (2) führen, angewendet werden. Solche Kanten werden als *Regresskanten* bezeichnet. Die verallgemeinerte Sweep-Line-Methode funktioniert nun wie folgt: Sobald eine Regresskante während eines Durchlaufs der Methode auftritt, wird ihr Zielzustand als *persistent* gekennzeichnet; diese persistenten Zustände werden während des aktuellen Durchlaufs nicht weiter durchmustert, werden jedoch im Speicher gehalten. Ist ein Durchlauf beendet, wird eine neue Iteration mit den persistenten Zuständen als Menge von Startzuständen begonnen (in dieser Iteration können natürlich neue persistente Zustände auftreten, für die wiederum eine neue Iteration

<sup>1</sup>Die Definition ist bereits an unsere Zwecke angepasst; in [Mai03] wird die Progress Measure für beliebige partiell geordnete Zielmengen definiert.

## 2 Grundlagen

gestartet werden muss). So werden alle Zustände des Erreichbarkeitsgraphen mindestens einmal durchmustert; allerdings ist es möglich, dass Zustände mehrmals durchmustert werden. Dem folgenden Satz kann die maximale Anzahl der Iterationen des Sweep-Line-Algorithmus bezüglich eines bestimmten Systems entnommen werden:

**Satz 1 ([Mai03])** Sei  $s_i$  ein Zustand in einem Transitionssystem und  $\text{reach}(s_i)$  die Menge aller Zustände, die von  $s_i$  aus erreichbar sind. Wenn in einem Transitionssystem alle erreichbaren Zustände mit dem Durchlaufen von maximal  $n$  Regresskanten erreicht werden können, dann terminiert der Algorithmus nach  $n + 2$  Iterationen. Dabei hat er höchstens  $(n + 2) \cdot |\text{reach}(s_i)|$  Zustände durchmustert.

Die Anzahl der maximalen Iterationen des Sweep-Line-Algorithmus steigt also proportional zur Anzahl der Regresskanten eines Erreichbarkeitsgraphen.

### 2.3 Algebraische Berechnung von Progress-Werten

Möchte man die Sweep-Line-Methode komplett automatisiert für den Zustandsraum einer bestimmten formalen Methode anwenden, ist es nötig, eine automatische Berechnung der Progress-Werte bereitzustellen. In [Sch04] entwickelt der Autor eine Methode zur automatischen Berechnung von Progress-Werten für Petrinetze, so dass die Sweep-Line-Methode auf Erreichbarkeitsgraphen von Petrinetzen angewendet werden kann. Die Methode wurde bereits erfolgreich in das Model-Checking-Tool LoLA [Sch00] integriert.

Im Folgenden wollen wir die Funktionsweise der Methode erläutern.

Ziel ist es zunächst, jeder Transition einen *Offset-Wert*  $o(t_i)$  zuzuordnen; die Offset-Werte der einzelnen Transitionen werden später zu Progress-Werten kombiniert.

Im ersten Schritt wird dazu die größte Menge linear unabhängiger Transitionsvektoren bestimmt, welche mit  $U$  bezeichnet wird (folglich gilt  $|U| = r(M)$ ). Fasst man die Transitionsvektoren  $\Delta t_i$  als Elemente eines Vektorraums auf, bildet  $U$  die Basis dieses Vektorraumes; daher bezeichnen wir  $U$  im Folgenden auch kurz als *Basis*. Jede Transition  $t_i$ , deren Vektor in  $U$  enthalten ist, erhält den Offset-Wert  $o(t_i)=1$ . Die Transitionsvektoren, die nicht in  $U$  enthalten sind, können durch die Linearkombination  $\Delta t_j = \lambda_0 \Delta t_0 + \dots + \lambda_n \Delta t_n$  mit  $U = t_0, \dots, t_n$  beschrieben werden. Jede der entsprechenden Transitionen erhält deshalb den Offset-Wert  $o(t_j) = \lambda_1 o(t_0) + \dots + \lambda_n o(t_n)$ . Transitionen mit negativem Offset-Wert werden als *Regresstransitionen* bezeichnet, da jede Aktivierung einer solchen Transition eine Regresskante im Erreichbarkeitsgraphen des Petrinetzes zur Folge hat.

Nun kann die Progress Measure des Erreichbarkeitsgraphen eines Petrinetzes auf Grundlage der Offset-Werte berechnet werden.

Sei  $p(s)$  eine Progress Measure bezüglich einer Markierung  $s$  eines Petrinetzes. Für die Anfangsmarkierung  $s_0$  eines Petrinetzes wird  $p(s_0) = 0$  gesetzt. Die Progress Measure wird nun als inkrementelles Maß berechnet: Für jede Nachfolgermarkierung  $s'$  der Markierung  $s$ , die durch das Schalten der Transition  $t_i$  zustande kommt, gilt  $p(s') = p(s) + o(t_i)$ .

Durch die Berechnung wird gewährleistet, dass einer Markierung auch dann ein eindeutiger Progress-Wert zugewiesen wird, wenn sie über verschiedene Schaltsequenzen erreichbar ist.

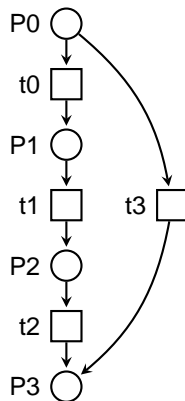


Abbildung 2.3: Beispiel-Petrinetz zur ursprünglichen Berechnung der Offset-Werte

### 2.3.1 Beispiel

Das folgende Beispiel soll die eben vorgestellte Berechnung der Offset-Werte demonstrieren. Gegeben sei das Petrinetz aus Abb. 2.3, welches durch die Inzidenzmatrix

$$M = \begin{pmatrix} -1 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

definiert ist. Wählt man  $U = \{t_0, t_1, t_2\}$  als Menge linear unabhängiger Transitionen, gilt  $o(t_0) = o(t_1) = o(t_2) = 1$ . Nun muss  $o(t_3)$  berechnet werden. Der Vektor der Transition  $t_3$  kann durch  $\Delta t_3 = \Delta t_0 + \Delta t_1 + \Delta t_2$  als Linearkombination der linear unabhängigen Transitionsvektoren dargestellt werden. Somit folgt  $o(t_3) = o(t_0) + o(t_1) + o(t_2)$ , also  $o(t_3) = 3$ .

Wir können jedoch beispielsweise auch  $U = \{t_1, t_2, t_3\}$  wählen. Dann erhalten wir  $o(t_1) = o(t_2) = o(t_3) = 1$  sowie  $o(t_0) = -o(t_1) - o(t_2) + o(t_3) = -1$ .

### 2.3.2 Geometrische Interpretation

In [Sch04] erwähnt der Autor, dass die Berechnung der Progress-Werte auch geometrisch interpretiert werden kann: Man betrachte den euklidischen Raum  $\mathbb{Q}^{|P|}$ . In diesem Raum bildet jede Markierung einen Punkt; die Transitionsvektoren  $\Delta t_i$  werden als Vektoren in diesem Raum betrachtet. Sei nun  $s$  eine Markierung und  $t_i$  eine Transition. Das Schalten von  $t_i$  entspricht dann der Verschiebung von  $s$  um  $\Delta t_i$ , es ergibt sich also wie gehabt  $s' = s + \Delta t_i$ . Ebenso kann man die linear unabhängigen Transitionen als Punkte in diesem Raum auffassen, nämlich als Translation des Punktes  $\underline{0}$  durch den entsprechenden Vektor. Diese Punkte definieren eine Hyperebene  $E$ ; dies ist die minimale Hyperebene, die alle durch die linear unabhängigen Transitionen definierten Punkte enthält. Aus dieser Interpretation kann, wie in [Sch04] beschrieben, die Progress Measure einer Markierung  $s$  abgeleitet werden: Sei  $d$  der Punkt der Hyperebene mit dem kleins-

## 2 Grundlagen

ten Abstand zu  $\underline{0}$  und  $g$  die Gerade, die durch  $d$  und  $\underline{0}$  verläuft. Sei nun  $i$  der Schnittpunkt von  $g$  mit der zu  $E$  parallelen Hyperebene, die  $s$  enthält. Dann ist der Progress-Wert von  $s$  der Abstand zwischen  $i$  und  $\underline{0}$ , wobei der Abstand zwischen  $\underline{0}$  und  $d$  den Progress-Wert 1 definiert.

### 2.3.3 Ansätze zur Optimierung

Die vorgestellte Berechnungsmethode lässt einige Optimierungsmöglichkeiten zu, die wir im Folgenden betrachten wollen.

1. Da nach Satz 1 die Anzahl der maximalen Iterationen der Sweep-Line-Methode proportional zur Anzahl der Regresskanten des Erreichbarkeitsgraphen steigt, ist es sinnvoll, *die Anzahl der Regresstransitionen zu minimieren*. Zudem wird sich durch solch eine Minimierung in vielen Fällen (abhängig vom jeweiligen Petrinetz) die Anzahl der persistenten Zustände des Erreichbarkeitsgraphen, welche im Speicher zu halten sind, reduzieren, da jede Aktivierung einer Regresstransition eine Regresskante im Erreichbarkeitsgraphen und somit einen möglicherweise zusätzlichen persistenten Zustand zur Folge hat. Das Ziel der nachfolgenden Kapitel ist es, einen Ansatz zur Minimierung der Anzahl an Regresstransitionen zu erarbeiten.

2. Weitere Verbesserungen könnten erzielt werden, indem versucht wird, *Ketten von Regresstransitionen zu vermeiden*. Diese können dazu führen, dass in einer Vielzahl von Iterationen unnötigerweise immer wieder dieselben Zustände durchmustert werden. Insbesondere bei einer langen Kette von Regresstransitionen besteht die Gefahr, dass der Zielzustand dieser Kette weit hinter der Front liegt und somit große Teile des Erreichbarkeitsgraphen erneut durchmustert werden.

3. Des Weiteren existiert der Ansatz der *Sweep-Bar-Methode* als Modifikation der Sweep-Line-Methode. Der Unterschied zur Sweep-Line-Methode besteht darin, dass eine gewisse Anzahl der bereits inklusive aller Folgezustände durchmusterten Zustände noch im Speicher gehalten wird, so dass man sich die Front eher als einen „Balken“ als als eine „Linie“ vorstellen kann – einige Zustände, die bisher der Klasse (2) zuzuordnen waren, gehören jetzt zur Klasse (3). Befindet sich nun der Zielzustand einer soeben entdeckten Regresskante innerhalb der Klasse (3) (also innerhalb der Sweep-Bar), muss dieser nicht als persistent markiert werden und ist somit nicht Ausgangspunkt einer neuen Iteration. Demnach kann die Anzahl der Iterationen durch die Sweep-Bar im Vergleich zur Sweep-Line reduziert werden, sofern Regresskanten innerhalb der Sweep-Bar verlaufen. Die Schwierigkeit dieses Ansatzes besteht darin, die optimale Anzahl der zusätzlich zur ursprünglichen Sweep-Line im Speicher zu haltenden Zustände zu ermitteln, um einerseits möglichst viele Iterationen einzusparen, andererseits aber nicht zu viel Speicherplatz zu verbrauchen. Möglicherweise kann die optimale Größe der Sweep-Bar aus den Regresstransitionen abgeleitet werden.

Grundsätzlich existieren zwei Freiheitsgrade in der beschriebenen Berechnungsvorschrift, die optimiert werden können: Die Zusammensetzung der Menge  $U$  sowie die Offset-Werte der linear unabhängigen Transitionen. Sind alle Transitionsvektoren linear unabhängig, ist  $U$  eindeutig bestimmt; alle Transitionen erhalten den Offset-Wert 1, somit muss nichts optimiert werden, da die Sweep-Line-Methode den kompletten Erreichbarkeitsgraphen in einem Durchlauf durchmustern

### 2.3 Algebraische Berechnung von Progress-Werten

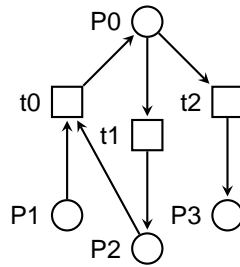


Abbildung 2.4: Ein Petrinetz, bei dem keine Optimierung nötig ist

kann. Solch ein Netz ist in Abb. 2.4 dargestellt. Abhängig vom Netz kann es jedoch auch möglich sein, dass verschiedene maximale Mengen linear unabhängiger Vektoren existieren. Eine geeignete Wahl dieser Menge könnte sowohl zur Minimierung der Anzahl negativer Offset-Werte als auch zur Vermeidung von Ketten von Regresstransitionen beitragen. Nach der Wahl der Basis könnten dann die Offset-Werte der linear unabhängigen Transitionen optimiert werden. Bisher bekommt jede linear unabhängige Transition  $t_i$  den Offset-Wert  $o(t_i) = 1$  zugewiesen; allerdings führt jeder beliebige Wert  $o(t_i) \neq 0$  für diese Transitionen zu einer konsistenten Progress Measure. Somit könnte auch die Veränderung dieser Offset-Werte zu Verbesserungen bei den unter **1.** und **2.** aufgeführten Optimierungsmöglichkeiten führen.

Insbesondere bei der Wahl der Basis treten allerdings Schwierigkeiten auf. Da es zu aufwändig wäre, zuerst alle maximalen Mengen linear unabhängiger Vektoren zu berechnen und anschließend die Menge auszuwählen, durch welche die geringste Anzahl an negativen Offset-Werten berechnet wird, müsste eine geeignete Heuristik zur Wahl der Basis gefunden werden.

Da wir die Entwicklung einer passenden Heuristik nicht erfolgreich abschließen konnten, soll im folgenden Kapitel eine neuer, auf der in [Sch04] entwickelten Methode basierender Ansatz zur automatischen Berechnung der Offset-Werte vorgestellt werden, der uns für eine Optimierung besser geeignet erscheint.



### 3 Geometrisch basierter Ansatz zur Berechnung der Offset-Werte

Der neue Ansatz zur Berechnung der Offset-Werte basiert auf der in 2.3.2 beschriebenen geometrischen Interpretation der Berechnung der Progress-Werte sowie folgender Feststellung über die von den linear unabhängigen Transitionsvektoren aufgespannte Hyperebene  $E$  in [Sch04]:

*„An optimal position would be such that as many as possible transitions point to the same side of the parallel of  $E$  through point  $\underline{0}$ .“*

In Abb. 3.1 ist eine optimale Lage von  $E$  für ein Beispiel im zweidimensionalen Raum  $\mathbb{Q}^{|\mathcal{P}|}$  dargestellt.

Wir werden nun genauer betrachten, wie die Offset-Werte mit einer durch den Nullpunkt verlaufenden Hyperebene in Zusammenhang stehen.

Gegeben sei ein Petrinetz in Form seiner Inzidenzmatrix  $M$ . Die Offset-Werte der linear abhängigen Transitionen ergeben sich bisher aus der Linearkombination der Offset-Werte der linear unabhängigen Transitionen ( $o(t_j) = \lambda_0 o(t_0) + \dots + \lambda_n o(t_n)$  wie oben beschrieben). Sie können allerdings auch als Progress-Werte aufgefasst werden, die durch das  $\lambda_i$ -malige Schalten der einzelnen Transitionen  $t_i$  entstehen. Betrachten wir daher nun nicht nur die linear unabhängigen Transitionen, sondern alle Transitionen als Punkte im Raum  $\mathbb{Q}^{|\mathcal{P}|}$ . Weiterhin sei  $E$  eine Hyperebene, die durch den Punkt  $\underline{0}$  verläuft. Hyperebenen können eindeutig durch einen orthogonal zu ihr stehenden Vektor beschrieben werden. Sei nun  $a$  ein orthogonal zu  $E$  stehender Vektor, der seinen Ursprung im Punkt  $\underline{0}$  hat. Jetzt können wir  $E$  durch

$$E = \{x \mid a^T x = 0\} \quad (3.1)$$

als Menge aller Punkte beschreiben, die in  $E$  enthalten sind.  $E$  teilt den Raum  $\mathbb{Q}^{|\mathcal{P}|}$  in zwei Halbräume. Auch diese können wir, analog zur Hyperebene, als Menge der in ihnen enthaltenen Punkte definieren:

$$\{x \mid a^T x > 0\} \quad (3.2)$$

sowie

$$\{x \mid a^T x < 0\} \quad (3.3)$$

Für einen Transitionsvektor  $\Delta t_i$  gilt offensichtlich  $a \cdot \Delta t_i > 0$ , wenn er in den selben Halbraum zeigt wie der Vektor  $a$  und  $a \cdot \Delta t_i < 0$ , wenn er in den anderen Halbraum zeigt. Zeigt der Transitionsvektor auf die Hyperebene, gilt  $a \cdot \Delta t_i = 0$ . Man kann also für eine gegebene Inzidenzmatrix  $M$  den Vektor  $x = a^T M$  berechnen, um festzustellen, welche Transitionen in welchem Halbraum

### 3 Geometrisch basierter Ansatz zur Berechnung der Offset-Werte

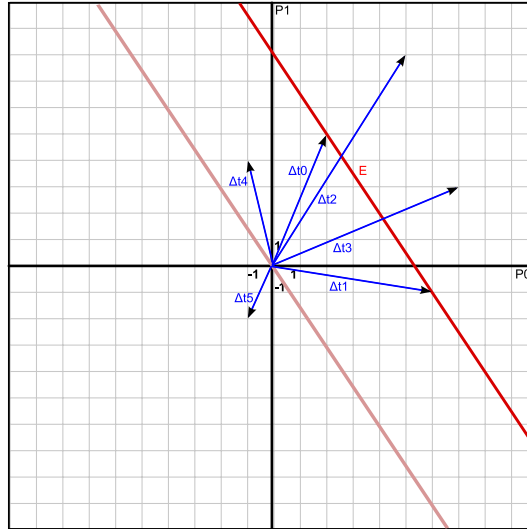


Abbildung 3.1: Ein Beispiel für die optimale Lage einer Hyperebene  $E$  im Zweidimensionalen

bzw. auf der Hyperebene liegen.

Der folgende Satz zeigt, wie diese Betrachtungsweise für die Berechnung der Offset-Werte genutzt werden kann:

**Satz 2** Sei  $M = |P| \times |T|$  die Inzidenzmatrix eines Petrinetzes und  $a$  ein Vektor der Dimension  $|P|$  mit dem Ursprung  $\underline{0}$ . Sind alle Komponenten  $x_i$  des Vektors  $x = a^T M$  ungleich 0, dann gilt:  $x_i$  ergibt den Offset-Wert der Transition  $t_i$ .

Wir werden nun skizzieren, weshalb dieser Satz gilt. Sei  $U$  eine Menge maximal linear unabhängiger Transitionsvektoren des gegebenen Petrinetzes. Diese Vektoren spannen eine Hyperebene  $E$  auf, die den Punkt  $\underline{0}$  nicht enthält. Mit  $E'$  bezeichnen wir nun die Hyperebene, welche parallel zu  $E$  durch den Punkt  $\underline{0}$  verläuft. Nun lässt sich ein zu  $E'$  orthogonal stehender Vektor  $a$  mit dem Ursprung  $\underline{0}$  finden, der auf  $E$  zeigt. Wir haben also zwei Möglichkeiten gefunden, um  $E$  zu beschreiben: durch die Menge  $U$  von Vektoren oder durch den einzelnen Vektor  $a$ . Dieser Vektor hat, da er orthogonal zu  $E$  steht, als Länge den kürzesten Abstand zwischen  $E$  und  $\underline{0}$ . Auf der jetzt beschriebenen Hyperebene liegen alle Transitionen, die den Offset-Wert 1 besitzen. Wie oben beschrieben, können wir die Offset-Werte der linear unabhängigen Transitionen auch als Progress-Werte auffassen. Um also den Offset-Wert einer linear abhängigen Transitionen  $t_i$  zu bestimmen, können wir uns einen Vektor  $a'$  vorstellen, der dieselbe Lage besitzt wie  $a$ , jedoch in der Länge so verändert ist, dass er genau auf die zu  $E$  parallele Hyperebene zeigt, die  $t_i$  enthält; der Offset-Wert dieser Transition ist dann die Länge von  $a'$ . Genau dies wird durch  $x_i = a^T \Delta t_i$  angegeben. Die Offset-Werte aller Transitionen können daher mit  $x = a^T M$  berechnet werden.

Die Länge von  $a$  beeinflusst unsere Berechnung insofern, als dass sich die Größe der Offset-Werte, nicht jedoch ihr Verhältnis zueinander ändert. Multipliziert man  $a$  mit einem Faktor  $k \in \mathbb{Q}$ , dann verändert sich auch jeder Offset-Wert  $o(t_i)$  um diesen Faktor  $k$ .

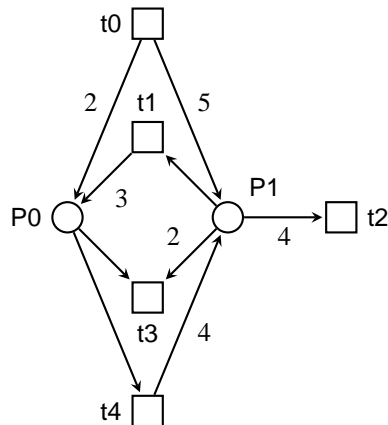


Abbildung 3.2: Beispiel-Petrinetz zur geometrisch basierten Berechnung der Offset-Werte

Durch unsere neue Methode existiert statt wie bisher zwei Freiheitsgraden (der Wahl der Basis und der Veränderung der Offset-Werte) mit dem Vektor  $a$  nur noch eine multidimensionale Variable, die verändert werden kann, um eine möglichst geringe Anzahl an negativen Offset-Werten zu erreichen. Zudem muss die (noch nicht zum Punkt  $\underline{0}$  verschobene) Hyperebene jetzt nicht mehr zwangsläufig linear unabhängige Vektoren enthalten. Die durch  $\underline{0}$  verlaufende Hyperebene kann nun (bis auf einige Einschränkungen, wie wir später sehen werden) beliebig gedreht werden. Hierdurch kann die Anzahl an Regresstransitionen möglicherweise in größerem Maße verringert werden als mit der vorherigen Einschränkung.

### 3.1 Beispiel zur Veranschaulichung

Im folgenden soll die eben beschriebene Methode zur automatischen Berechnung von Offset-Werten anhand eines Beispiels veranschaulicht werden. Gegeben sei ein Petrinetz mit der Inzidenzmatrix

$$M = \begin{pmatrix} 2 & 3 & 0 & -1 & -1 \\ 5 & -1 & -4 & -2 & 4 \end{pmatrix}$$

; dieses ist in Abb. 3.2 dargestellt. Die 5 Transitionen des Netzes betrachten wir nun als Punkte im Raum  $\mathbb{Q}^{|P|}$ ; da das Netz nur 2 Plätze enthält, können wir uns dies in einem zweidimensionalen Koordinatensystem darstellen (siehe Abb. 3.3). Nun wählen wir den Vektor  $a^T = (5, 2)$ , der die Hyperebene  $E$  beschreibt. Dieser teilt den Raum  $\mathbb{Q}^{|P|}$  in die Halbräume  $\{x \mid a^T x > 0\}$  sowie  $\{x \mid a^T x < 0\}$  (siehe Abb. 3.4). Jetzt berechnen wir  $x = a^T M = (20, 13, -8, -9, 3)$ . Wie an Abb. 3.4 zu erkennen ist, liegen genau die positiven Komponenten von  $x$  in dem Halbraum, in den  $a$  zeigt. Da keine Komponente des Vektors  $x$  den Wert 0 annimmt, bilden sie gültige Offset-Werte für die einzelnen Transitionen; es gilt  $o(t_i) = x_i$ , also  $o(t_0) = 20$ ,  $o(t_1) = 13$ ,  $o(t_2) = -8$ ,  $o(t_3) = -9$  und  $o(t_4) = 3$ .

### 3 Geometrisch basierter Ansatz zur Berechnung der Offset-Werte

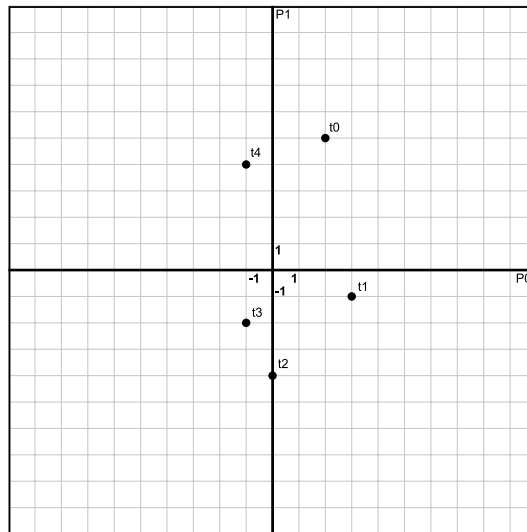


Abbildung 3.3: Darstellung der Transitionen aus Abschnitt 3.1 im Koordinatensystem

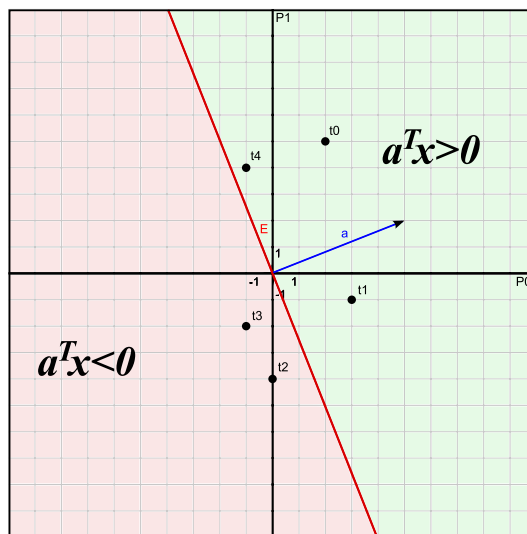


Abbildung 3.4: Darstellung der durch  $a$  definierten Halbräume

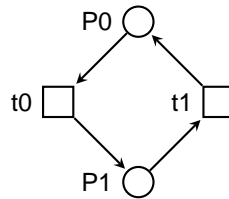


Abbildung 3.5: Ein Petrinetz, bei dem Probleme mit dem Offset-Wert 0 auftreten

### 3.2 Probleme mit Nullen als Offset-Werte

Durch die Berechnung der Offset-Werte nach der algebraischen Methode war sichergestellt, dass linear unabhängige Transitionen den Offset-Wert 1 erhalten; durch den neuen, geometrisch basierten Ansatz zur Berechnung können diese Transitionen jeden beliebigen Offset-Wert annehmen. Sofern die Offset-Werte aller Transitionen ungleich 0 sind, ergibt sich trotzdem eine konsistente Progress Measure und die Sweep-Line-Methode kann ohne Probleme ausgeführt werden. Es ist jedoch nach dem geometrisch basierten Ansatz auch möglich, dass voneinander linear abhängige Transitionen den Offset-Wert 0 erhalten, wenn wir die Wahl des Vektors  $a$  nicht einschränken. Schauen wir uns hierzu das folgende Beispiel an:

Sei

$$M = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$$

die Inzidenzmatrix eines Petrinetzes mit den Plätzen  $P = \{P_0, P_1\}$  und den Transitionen  $T = \{t_0, t_1\}$ . Das zugehörige Petrinetz ist in Abb. 3.5 dargestellt. Wie man an der Inzidenzmatrix erkennen kann, sind beide Transitionen voneinander linear abhängig; nach der Vorgehensweise der algebraischen Methode kann man entweder  $t_1$  oder  $t_2$  als linear unabhängige Transition wählen. Wählt man  $t_1$  als linear unabhängig, erhält man  $o(t_0) = 1$  und  $o(t_1) = -o(t_0) = -1$ ; wählt man  $t_1$  als linear unabhängige Transition, erhält man analog  $o(t_1) = 1$  und  $o(t_0) = -o(t_1) = -1$ . Anhand dieser Werte wird deutlich, was auch an Abb. 3.5 zu erkennen ist: Eine der beiden Transitionen sorgt für „Fortschritt“, während das Schalten der anderen Transition einen „Rückschritt“ bedeutet.

Möchte man die Offset-Werte nach der geometrisch basierten Methode berechnen, muss der Vektor  $a^T = (a_1, a_2)$  angegeben werden. Berechnet man nun  $x = a^T M$ , erhält man genau dann  $x = (0, 0)$ , wenn  $a_1 = a_2$  gilt. Da nun alle Transitionen den Offset-Wert 0 besitzen, ist auch der Progress-Wert für alle Markierungen 0. Weil alle Markierungen den gleichen Progress-Wert haben, ist nicht mehr erkennbar, welche Zustände aus dem Speicher gelöscht werden können - die Sweep-Line-Methode wird hinfällig. Es muss also verhindert werden, dass Transitionen, die voneinander linear abhängig sind, den Offset-Wert 0 erhalten. Die einfachste Lösung dieses Problems ist es, nur Vektoren  $a$  zuzulassen, die für alle Transitionen zu Offset-Werten führen, welche ungleich 0 sind. Geometrisch interpretiert bedeutet dies, dass kein Transitionsvektor auf die gewählte Hyperebene zeigen darf.

Da nach der Berechnungsmethode des algebraischen Ansatzes Transitionen den Offset-Wert 0 erhalten können, ohne dass die Konsistenz der Progress Measure dadurch beeinträchtigt ist, ist

### *3 Geometrisch basierter Ansatz zur Berechnung der Offset-Werte*

es offensichtlich, dass es bessere Lösungen für dieses Problem geben muss, in denen Nullen als Offset-Werte nach gewissen Kriterien zugelassen sind. Das Auffinden einer solchen Lösung ist nicht Bestandteil dieser Arbeit. Wir wenden uns stattdessen der Optimierung der Anzahl negativer Offset-Werte zu.

## 4 Optimierung der Anzahl negativer Offset-Werte

In Abschnitt 2.3.3 wurde gezeigt, dass sich die Minimierung der Anzahl negativer Offset-Werte positiv auf die Laufzeit der Sweep-Line-Methode auswirken kann. Aus der im vorherigen Abschnitt beschriebenen Berechnungsmethode kann jetzt ein Ansatz zur Optimierung der Offset-Werte entwickelt werden.

Um möglichst wenige negative Offset-Werte (und damit wenige Regresstransitionen) zu erhalten, muss eine durch  $\underline{0}$  verlaufende Hyperebene gefunden werden, die den Raum  $\mathbb{Q}^{|P|}$  so in zwei Halbräume teilt, dass möglichst viele Transitionen in einem und somit möglichst wenige Transitionen im anderen Halbraum liegen. Gesucht wird also ein Vektor  $a$  so, dass der Vektor  $x = a^T M$  möglichst viele positive Komponenten enthält. (Ebenso könnte ein  $a$  so gesucht werden, dass  $x$  möglichst viele negative Komponenten enthält; hat man dieses gefunden, muss  $x = -a^T M$  berechnet werden, um wiederum ein  $x$  mit möglichst vielen positiven Komponenten zu erhalten.)

Folgender Ansatz bringt uns dem Ziel, ein solches  $a$  zu finden, ein großes Stück näher:

Gegeben sei die Inzidenzmatrix

$$M = \begin{pmatrix} m_{11} & \cdots & m_{1k} \\ \cdots & \cdots & \cdots \\ m_{n1} & \cdots & m_{nk} \end{pmatrix}$$

eines Petrinetzes mit den Transitionen  $T = \{t_0, \dots, t_{k-1}\}$  sowie den Plätzen  $P = \{p_0, \dots, p_{n-1}\}$ . Weiterhin sei  $a^T = (a_1, \dots, a_n)$  ein Vektor. Die Forderung, dass möglichst viele Komponenten des Vektors  $x = a^T M$  größer als 0 sein sollen, lässt sich so auffassen, dass von den folgenden  $k$  Ungleichungen möglichst viele erfüllt werden müssen:

$$\begin{aligned} m_{11} \cdot a_1 + \cdots + m_{n1} \cdot a_n &> 0 \\ \cdots + \cdots + \cdots &> 0 \\ m_{1k} \cdot a_1 + \cdots + m_{nk} \cdot a_n &> 0 \end{aligned}$$

Der Transition  $t_{i-1}$  ist also die Ungleichung

$$m_{1i} \cdot a_1 + \cdots + m_{ni} \cdot a_n > 0 \tag{4.1}$$

zugeordnet. Die linke Seite der Ungleichung entspricht hierbei dem Offset-Wert der jeweiligen Transition. Diese Ungleichungen betrachten wir nun als Nebenbedingungen eines linearen Optimierungsproblems (im Folgenden auch kurz als  $LP$  für „Lineares Programm“ bezeichnet). Sind alle Ungleichungen erfüllbar, ist das  $LP$  zulässig; die Anzahl negativer Offset-Werte kann

#### 4 Optimierung der Anzahl negativer Offset-Werte

zu 0 minimiert werden, so dass das zugehörige Petrinetz keine Regresstransitionen enthält. Als Vektor  $a$  kann dann ein beliebiger Wert aus der Menge der zulässigen Lösungen des LPs gewählt werden, um nur nichtnegative Offset-Werte zu erhalten. Bei vielen Petrinetzen werden sich allerdings nicht alle Regresstransitionen eliminieren lassen. In solchen Fällen ist es nicht möglich, dass alle Ungleichungen gleichzeitig erfüllt sind; das zugehörige LP ist unzulässig. Somit kann kein Vektor  $a$  gefunden werden, so dass  $x$  nur positive Komponenten enthält. Da die Anzahl negativer Offset-Werte aber nur vor der Berechnung von  $x$  bestimmt werden kann, wenn  $a$  aus einer zulässigen Menge eines LPs gewählt wird, versuchen wir nun, aus der obigen Menge von Ungleichungen eine Teilmenge zu finden, die wiederum eine zulässige Menge eines LPs beschreibt. Hierbei ist es unser Ziel, möglichst wenige Ungleichungen zu eliminieren, da für Transitionen, deren zugehörige Ungleichung (4.1) eliminiert wird, nicht garantiert ist, dass ihr Offset-Wert größer als 0 ist. Je weniger Ungleichungen eliminiert werden, desto mehr Transitionen kann demnach garantiert ein nichtnegativer Offset-Wert zugewiesen werden. Wir suchen also aus unserer Menge von Ungleichungen die größtmögliche Teilmenge, die Nebenbedingung eines zulässigen Linearen Programms sein kann. Auf dieses Problem wollen wir im folgenden Abschnitt näher eingehen.

### 4.1 Das Maximum Feasible Subsystem Problem

Das *Maximum Feasible Subsystem Problem* (kurz MAXFS) beschäftigt sich mit der Frage, wie aus einer Menge unzulässiger linearer Nebenbedingungen eine zulässige Teilmenge maximaler Größe gewonnen werden kann. Eine exakte Formulierung des Problems findet man in [APT99]:

**Definition 2 (MAXFS)** Sei  $\Sigma : \{Ax \leq b\}$  ein unzulässiges System mit  $A \in \mathbb{R}^{m \times n}$  und  $b \in \mathbb{R}^m$ . Finde ein zulässiges Teilsystem, das so viele Ungleichungen wie möglich enthält.

Das Problem kann auch derart betrachtet werden, dass die kleinste Menge an Ungleichungen gesucht wird, welche aus einem unzulässigen System entfernt werden muss, um ein zulässiges zu erhalten; dann wird es als *Minimum Unsatisfied Linear Relation Problem* (MINULR) [AK94] bezeichnet. Es existiert noch eine weitere Betrachtungsweise, für die wir zunächst folgende Definition benötigen:

**Definition 3 (IIS)** Eine irreduzible, unzulässige Menge von Nebenbedingungen (kurz IIS für „irreducible infeasible set“) ist eine unzulässige Menge von Nebenbedingungen, von der jede echte Teilmenge eine Menge zulässiger Nebenbedingungen bildet.

Somit muss aus einem IIS nur eine Nebenbedingung entfernt werden, um ein zulässiges System zu erhalten. Eine Menge unzulässiger Nebenbedingungen kann eine exponentielle Anzahl an IIS enthalten [Cha94]. Nun kann man versuchen, die Menge mit der kleinsten Anzahl an Ungleichungen zu finden, so dass diese mindestens eine Ungleichung aus jedem IIS des betrachteten Systems enthält; dieses Problem ist bekannt als *Minimum-Cardinality IIS Set Covering Problem* (kurz MIN IIS COVER) [Chi96]. MIN IIS COVER ist identisch zu MINULR.

MAXFS ist NP-schwer [AK95, Cha94]; dementsprechend sind angepasste Algorithmen nötig, um MAXFS-Instanzen in vertretbarer Zeit exakt oder annäherungsweise zu lösen.

### 4.1.1 Algorithmische Lösungsmethoden

Für MAXFS wurden verschiedene algorithmische Lösungsmethoden entwickelt, über die an dieser Stelle ein kurzer Überblick gegeben werden soll.

MAXFS kann mittels gemischt-ganzzahliger Programmierung exakt gelöst werden, wobei eine möglichst große, positive Variable, das sogenannte *Big-M*<sup>1</sup>, eingeführt wird, um herauszufinden, welche Ungleichungen zur Lösung des Problems eliminiert werden müssen [GM91]. Die geeignete Wahl des *Big-M* kann hierbei sowohl bezüglich des Auffindens des Optimums als auch bezüglich der Numerik (bei der Lösung des gemischt-ganzzahligen Programmes) Schwierigkeiten bereiten. [ABC08]

Zudem kann MAXFS als Lineares Programm mit Gleichgewichtsnebenbedingungen (*Linear Program with Equilibrium Constraints*, kurz: LPEC) exakt formuliert werden [Ama03]. Aufgrund der Nichtlinearität solcher Modelle ist es jedoch schwer, eine optimale Lösung zu finden.

Einige weitere Ansätze beziehen sich auf MIN IIS COVER. So wird beispielsweise vorgeschlagen, nach Aufzählung aller IIS ein Mengenüberdeckungsproblem über diesen zu lösen, um so eine Lösung für MIN IIS COVER zu erhalten [PR96]. Hierbei werden Heuristiken eingesetzt, um die Berechnung zu beschleunigen.

Eine auf diesem Ansatz sowie einem speziellen Branch-and-Cut-Algorithmus [CF04] basierende Methode liefert exakte Lösungen in kürzerer Zeit als alle anderen bisher existierenden exakten Methoden [Pfe08].

Eine Heuristik mit kurzer Laufzeit, die trotzdem für viele unzulässige Systeme sehr gute Resultate hervorbringt, wurde von Chinnek entwickelt [Chi01]. Innerhalb dieser Heuristik werden mehrere „elastische“ (*elastic*) Lineare Programme gelöst: Es werden zusätzliche „künstliche“ (*artificial*) Variablen eingeführt, die das LP zulässig werden lassen; die Lösung eines solchen LPs gibt Anhaltspunkte, welche Ungleichungen eliminiert werden müssen, um sich der Lösung eines MIN IIS COVER-Problems anzunähern.

In [ABC08] wird eine Zwei-Phasen-Heuristik vorgestellt; in der ersten Phase wird eine Relaxierung des MAXFS-Problems gelöst, welche in der zweiten Phase mittels einer exakten Formulierung des Problems verbessert werden soll.

Ferner wurden Algorithmen für approximative Lösungen von sehr großen Systemen mit hunderttausenden [MWE02] und Millionen [ABH05] von Ungleichungen entwickelt, welche für unsere Zwecke allerdings wohl kaum relevant sein werden, da Petrinetze mit einer solch hohen Anzahl an Transitionen in der Praxis kaum vorkommen dürften.

Aufgrund der Aktualität einiger genannter Publikationen kann davon ausgegangen werden, dass die Entwicklung von effizienten Algorithmen zur Lösung von MAXFS-Instanzen weiterhin Gegenstand der Forschung ist.

---

<sup>1</sup>Wir verwenden hier die in der Literatur gängige Bezeichnung; das *Big-M* steht in keiner Beziehung zur Inzidenzmatrix  $M$ .

#### 4 Optimierung der Anzahl negativer Offset-Werte

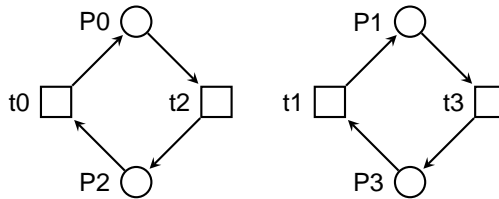


Abbildung 4.1: Ein Petrinetz, dessen Regresstransitionen nicht mit MAXFS minimiert werden können.

#### 4.1.2 Schwierigkeiten bei der Anwendung von MAXFS bezüglich der Optimierung

In Abschnitt 3.2 haben wir beschrieben, warum es problematisch ist, nach unserer geometrisch basierten Berechnungsmethode Nullen als Offset-Werte zuzulassen. Dieses Problem tritt nun wieder auf, wenn wir die Minimierung der Anzahl negativer Progress-Werte als MAXFS-Problem formulieren wollen. Nach Definition 2 müssen für ein MAXFS-Problem Ungleichungen der Form  $Ax \leq b$  gegeben sein; da wir jedoch Nullen als Offset-Werte vermeiden wollten, haben unsere Ungleichungen die Form  $Ax > 0$  (siehe S. 21). Das Problem, aus einer Menge von (beliebigen) unzulässigen linearen Relationen eine zulässige Teilmenge maximaler Größe zu gewinnen, wird als MAXFLS bezeichnet. Amaldi und Kann definieren das Problem  $\text{MAXFLS}^{\mathcal{R}}$  für  $\mathcal{R} \in \{=, \geq, >, \neq\}$  [AK95]. Im Gegensatz zu MAXFS konnten hierzu allerdings kaum Publikationen gefunden werden. Möglicherweise können jedoch von den genannten Algorithmen für MAXFS einige so modifiziert werden, dass mit ihnen MAXFLS-Probleme gelöst werden können. Betrachten wir an dieser Stelle ein Beispiel (siehe Abb. 4.1): Gegeben sei ein Petrinetz mit der Inzidenzmatrix

$$M = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

. Aus diesem können wir die Ungleichungen

$$a_1 - a_3 > 0 \tag{4.2}$$

$$a_2 - a_4 > 0 \tag{4.3}$$

$$-a_1 + a_3 > 0 \tag{4.4}$$

$$-a_2 + a_4 > 0 \tag{4.5}$$

ableiten. Es ist erkennbar, dass sowohl (4.2) und (4.4) als auch (4.3) und (4.5) nie gleichzeitig erfüllt sein können. Somit existieren für dieses System von Ungleichungen vier zulässige Teilsysteme maximaler Kardinalität:  $\{(4.2), (4.3)\}$ ,  $\{(4.2), (4.5)\}$ ,  $\{(4.3), (4.4)\}$  sowie  $\{(4.4), (4.5)\}$ . Möchten wir die Optimierung als MAXFS-Problem formulieren, müssen wir die  $\{>\}$ -Relationen durch  $\{\geq\}$ -Relationen ersetzen; dadurch wäre allerdings das gesamte System von Ungleichun-

## 4.2 Beispiel: Petrinetz ohne Regresstransitionen

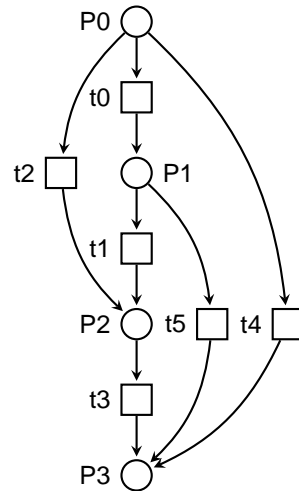


Abbildung 4.2: Ein Petrinetz, bei dem Regresstransitionen verhindert werden können

gen erfüllbar für  $a_1 = a_3$  und  $a_2 = a_4$ . Hierdurch würden alle Transitionen den Offset-Wert 0 erhalten; für dieses Petrinetz ist eine Formulierung als MAXFS-Problem demnach nicht sinnvoll.

## 4.2 Beispiel: Petrinetz ohne Regresstransitionen

Im Folgenden wollen wir ein Petrinetz betrachten, bei dem durch die oben beschriebene Betrachtungsweise Regresstransitionen verhindert werden können. Gegeben sei das durch die Inzidenzmatrix

$$M = \begin{pmatrix} -1 & 0 & -1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

definierte Petrinetz aus Abb. 4.2. Nach der Vorgehensweise der algebraischen Methode könnten wir jetzt beispielsweise die Basis  $U = \{t_0, t_3, t_4\}$  wählen; dann erhalten wir die Offset-Werte  $o(t_0) = o(t_3) = o(t_4) = 1$  sowie  $o(t_1) = o(t_4) - o(t_3) - o(t_0) = -1$ ,  $o(t_2) = o(t_4) + o(t_3) = 2$  und  $o(t_5) = o(t_4) - o(t_0) = 0$ . Wir erhalten durch diese Berechnung also eine Regresstransition, nämlich  $t_1$ . Berechnen wir die Offset-Werte jedoch nach unserem geometrisch basierten Ansatz aus Abschnitt 3, können wir wie oben beschrieben aus der Inzidenzmatrix sechs Ungleichungen von der Form (4.1) (siehe S. 21) ableiten:

$$\begin{aligned} -a_1 + a_2 &> 0 \\ -a_2 + a_3 &> 0 \\ -a_1 + a_3 &> 0 \\ -a_3 + a_4 &> 0 \end{aligned}$$

#### 4 Optimierung der Anzahl negativer Offset-Werte

$$\begin{aligned}-a_1 + a_4 &> 0 \\ -a_2 + a_4 &> 0\end{aligned}$$

Dieses System von Ungleichungen bildet eine Menge zulässiger Nebenbedingungen für ein Lineares Programm; somit können wir jeden beliebigen Punkt innerhalb der durch diese Ungleichungen beschriebene Menge als Vektor  $a$  für die Berechnung der Offset-Werte nutzen, um nur positive Offset-Werte zu erhalten. Wählen wir beispielsweise  $a^T = (1, 2, 3, 4)$ , erhalten wir  $x = a^T M = (1, 1, 2, 1, 3, 2) = (o(t_0), \dots, o(t_5))$ .

### 4.3 Beispiel: Petrinetz mit Regresstransitionen und die geometrische Interpretation von MaxFS

An dieser Stelle wollen wir noch einmal das Beispiel aus Abschnitt 3.1 aufgreifen, um zu veranschaulichen, warum man die Minimierung der Anzahl negativer Offset-Werte für dieses Petrinetz als MaxFS-Problem auffassen kann. Auch für dieses Beispiel können wir wiederum ein System von Ungleichungen erstellen:

$$\begin{aligned}2a_1 + 5a_2 &> 0 \\ 3a_1 - a_2 &> 0 \\ -4a_2 &> 0 \\ -a_1 - 2a_2 &> 0 \\ -a_1 + 4a_2 &> 0\end{aligned}$$

Es existiert kein Vektor  $a$ , für den alle Ungleichungen erfüllt sind; somit enthält das Petrinetz mindestens eine Regresstransition. Daher suchen wir nun eine größte Teilmenge dieser Ungleichungen, die ein zulässiges System bildet. Wir können uns das Problem für dieses Netz im Zweidimensionalen durch eine geometrische Interpretation veranschaulichen:

Jede der Ungleichungen unseres Systems beschreibt wiederum einen Halbraum; alle Punkte, die eine einzelne Ungleichung erfüllen, liegen in dem von ihr beschriebenen Halbraum. Stellen wir die Ungleichungen nun nach  $a_2$  um, so dass wir sie problemlos im Koordinatensystem darstellen können:

$$\begin{aligned}t_0 : a_2 &> -\frac{2}{5}a_1 \\ t_1 : a_2 &< 3a_1 \\ t_2 : a_2 &< 0 \\ t_3 : a_2 &< -\frac{a_1}{2} \\ t_4 : a_2 &> \frac{a_1}{4}\end{aligned}$$

#### 4.3 Beispiel: Petrinetz mit Regresstransitionen und die geometrische Interpretation von MAXFS

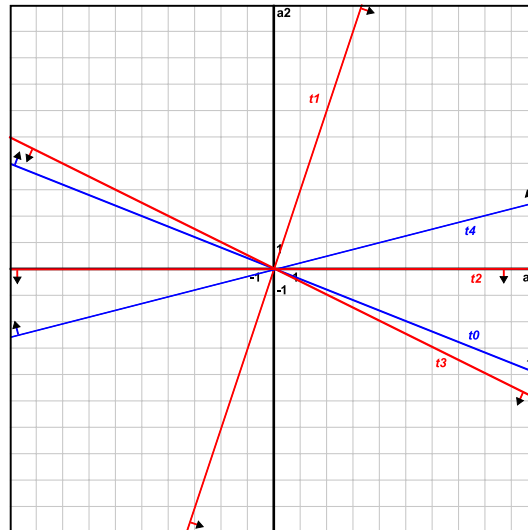


Abbildung 4.3: Darstellung der den Transitionen zugeordneten Halbräume

Die durch die Ungleichungen festgelegten Geraden sind in Abb. 4.3 dargestellt; für blaue Geraden liegt der beschriebene Halbraum oberhalb der Geraden, für rote Geraden liegt er unterhalb derselben (dies ist nochmals durch Pfeile dargestellt).

Der Schnitt mehrerer Halbräume bildet ein *Polyeder*. Wie in Abb. 4.3 zu erkennen ist, ist der Schnitt der dort dargestellten Halbräume leer. Wir suchen nun also nach der größtmöglichen Menge von Ungleichungen, die ein nichtleeres Polyeder beschreiben; dies ist die geometrische Interpretation des MAXFS-Problems<sup>2</sup>. Anhand der Abb. kann man erkennen, dass mindestens zwei Geraden entfernt werden müssen, um ein nichtleeres Polyeder zu erhalten. Entfernt man beispielsweise die zu  $t_2$  und  $t_3$  zugehörigen Geraden (und damit die entsprechenden Ungleichungen), wird durch die restlichen Ungleichungen ein nichtleeres Polyeder beschrieben (siehe Abb. 4.4). Diese Ungleichungen bilden nun das größte zulässige Teilsystem unseres unzulässigen Systems; dementsprechend besitzen alle Transitionen aus der Menge  $\{t_0, t_1, t_4\}$  positive Offset-Werte. Da es keine Menge von Ungleichungen mit größerer Kardinalität gibt, die ein zulässiges System beschreibt, existiert auch keine größere Menge an Transitionen mit positiven Offset-Werten. Aus der vom gefundenen Teilsystem beschriebenen Menge von Punkten kann man nun einen beliebigen Vektor  $a$  wählen (bspw.  $a^T = (5, 2)$  wie in Abschnitt 3.1), um Offset-Werte für die Transitionen des Petrinetzes zu berechnen, von denen maximal viele nichtnegativ sind.

<sup>2</sup>Genau genommen müssten die Ungleichungen für ein MAXFS-Problem wiederum als  $\{\leq\}$ -Relationen beschrieben werden, was in unserem Beispiel ohne Einschränkungen möglich wäre, da die Gleichheit hier hinzugefügt werden kann, ohne dass Probleme bei der hier vorgestellten Berechnung der Offset-Werte eintreten.

#### 4 Optimierung der Anzahl negativer Offset-Werte

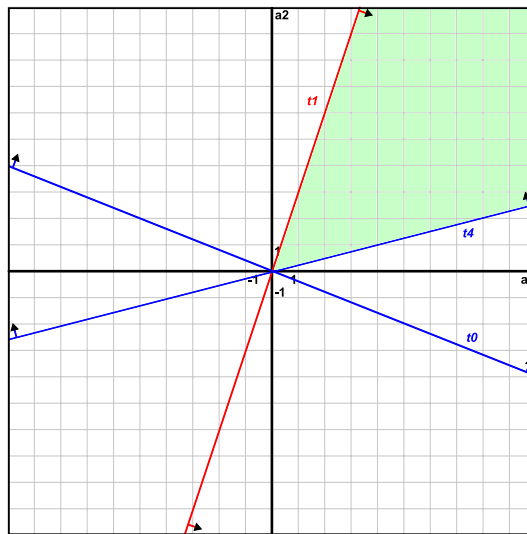


Abbildung 4.4: Darstellung des aus den Transitionen  $t_0$ ,  $t_1$  und  $t_4$  ableitbaren Polyeders (grün)

# 5 Schlussbemerkungen

## 5.1 Zusammenfassung

Das Ziel der vorliegenden Arbeit war es, einen Ansatz zur Optimierung der automatischen Berechnung von Progress-Werten für die Sweep-Line-Methode bei Petrinetzen zu erarbeiten, wobei wir uns auf die Minimierung der Anzahl negativer Offset-Werte konzentriert haben. Wir sind zunächst auf die Sweep-Line-Methode und auf eine bereits existierende algebraische Methode zur automatischen Berechnung von Progress-Werten für diese bei Petrinetzen eingegangen. Hierbei haben wir drei Optimierungsansätze beschrieben: Die Minimierung der Anzahl an Regresstransitionen, die Vermeidung von Ketten von Regresstransitionen sowie den Ansatz der Sweep-Bar-Methode. Aufbauend auf dem bisherigen Verfahren haben wir einen neuen, geometrisch basierten Ansatz zur Berechnung der Offset-Werte vorgestellt, welcher uns für eine Optimierung besser geeignet erscheint. Wir haben festgestellt, dass bei diesem Ansatz Probleme auftreten können, wenn Nullen als Offset-Werte zugelassen werden.

Im Anschluss haben wir einen Ansatz zur Optimierung der Anzahl negativer Offset-Werte entwickelt und haben festgestellt, dass unser Problem dem *Maximum Feasible Subsystem Problem* sehr ähnlich, wenn auch nicht zu diesem identisch ist; daher kann die Optimierung für einige, aber nicht alle Petrinetze als MAXFS-Problem aufgefasst werden.

## 5.2 Ausblick

In dieser Arbeit bleiben einige Fragen offen, die zukünftigen Arbeiten überlassen werden müssen.

Zunächst konnte nicht geklärt werden, nach welchen Kriterien in der Berechnung des geometrisch basierten Ansatzes der Offset-Wert 0 für eine Transition zugelassen werden kann, um eine konsistente Progress Measure zu erhalten. Dies stellt grundsätzlich eine Einschränkung dar, die möglicherweise für einige Petrinetze eine weitere Minimierung der Offset-Werte verhindert.

Zudem muss geprüft werden, ob existierende Algorithmen zur Lösung des MAXFS-Problems so abgewandelt werden können, dass sie ohne Einschränkung für unsere Optimierung genutzt werden können.

Wir haben in dieser Arbeit nur eine der in Abschnitt 2.3.3 angegebenen Optimierungsmöglichkeiten betrachtet. Nicht in allen Petrinetzen kann die Anzahl an Regresstransitionen minimiert werden; so treten beispielsweise im Petrinetz für den Mutex-Algorithmus <sup>1</sup> (siehe Abb. 5.1) immer genau zwei Regresstransitionen auf. Sowohl die Verhinderung von Ketten von Regresstransitionen als auch die Sweep-Bar-Methode können unabhängig von der Minimierung der Anzahl von Regresstransitionen betrachtet werden. Versuche, die Sweep-Bar-Methode ohne wei-

---

<sup>1</sup>Da dieses Netz einen konkreten Algorithmus modelliert, wurde hier eine Anfangsmarkierung mit angegeben.

## 5 Schlussbemerkungen

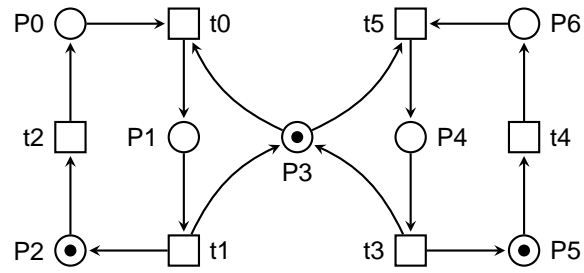


Abbildung 5.1: Petrinetz für den Mutex-Algorithmus

tere Optimierung zu nutzen, brachten allerdings nur einen sehr geringen Performancegewinn. Möglicherweise kann sie jedoch in Kombination mit den anderen beiden Optimierungskriterien vorteilhaft genutzt werden.

Das Ziel zukünftiger Arbeit sollte letztendlich sein, alle drei Optimierungskriterien so zu kombinieren, dass Progress-Werte berechnet werden, mit denen eine möglichst gute Performance der Sweep-Line-Methode erzielt werden kann.

# Literaturverzeichnis

- [ABC08] AMALDI, Edoardo ; BRUGLIERI, Maurizio ; CASALE, Giuliano: A two-phase relaxation-based heuristic for the maximum feasible subsystem problem. In: *Comput. Oper. Res.* 35 (2008), Nr. 5, S. 1465–1482
- [ABH05] AMALDI, Edoardo ; BELOTTI, Pietro ; HAUSER, Raphael: *Randomized relaxation methods for the maximum feasible subsystem problem*. Jünger, Michael (Hrsg.) et al., Integer programming and combinatorial optimization. 11th international IPCO conference, Berlin, Germany, June 8–10, 2005. Proceedings. Berlin: Springer. Lecture Notes in Computer Science 3509, S. 249-264, 2005
- [AK94] AMALDI, Edoardo ; KANN, Viggo: On the approximability of removing the smallest number of relations from linear systems to achieve feasibility. / Department of Mathematics, Swiss Federal Institute of Technology, Lausanne and Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm. 1994. – Forschungsbericht
- [AK95] AMALDI, Edoardo ; KANN, Viggo: The complexity and approximability of finding maximum feasible subsystems of linear relations. In: *Theor. Comput. Sci.* 147 (1995), Nr. 1-2, S. 181–210
- [Ama03] AMALDI, Edoardo: The maximum feasible subsystem problem and some applications. In: AGNETIS, A. (Hrsg.) ; PILLO, G. D. (Hrsg.): *Modelli e Algoritmi per l'Ottimizzazione di Sistemi Complessi*, 2003 (Pitagora Editrice Bologna), S. 31–69
- [APT99] AMALDI, Edoardo ; PFETSCH, Marc E. ; TROTTER, Leslie E.: *Some structural and algorithmic properties of the maximum feasible subsystem problem*. Cornuéjols, Gérard (Hrsg.) et al., Integer programming and combinatorial optimization. 7th international IPCO conference, Graz, Austria, June 9-11, 1999. Proceedings. Berlin: Springer. Lect. Notes Comput. Sci. 1610, S. 45-59, 1999
- [CF04] CODATO, Gianni ; FISCHETTI, Matteo: *Combinatorial Benders' cuts*. Bienstock, Daniel (Hrsg.) et al., Integer programming and combinatorial optimization. 10th international IPCO conference, New York, NY, USA, June 7–11, 2004. Proceedings. Berlin: Springer. Lecture Notes in Computer Science 3064, S. 178-195, 2004
- [Cha94] CHAKRAVARTI, Nilotpal: Some results concerning post-infeasibility analysis. In: *Eur. J. Oper. Res.* 73 (1994), Nr. 1, S. 139–143
- [Chi96] CHINNECK, John W.: An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem. In: *Ann. Math. Artif. Intell.* 17 (1996), Nr. 1-2, S. 127–144

## Literaturverzeichnis

- [Chi01] CHINNECK, John W.: Fast Heuristics for the Maximum Feasible Subsystem Problem. In: *INFORMS J. Comput.* 13 (2001), Nr. 3, S. 210–223
- [GM91] GREENBERG, Harvey J. ; MURPHY, Frederic H.: Approaches to diagnosing infeasible linear programs. In: *ORSA J. Comput.* 3 (1991), Nr. 3, S. 253–261
- [Mai03] MAILUND, Thomas: *Sweeping the State Space. A Sweep-Line State Space Exploration Method.*, Department of Computer Science, University of Aarhus, Diss., 2003
- [MWE02] MELLER, Jaroslaw ; WAGNER, Michael ; ELBER, Ron: Maximum feasibility guideline in the design and analysis of protein folding potentials. In: *Journal of Computational Chemistry* 23 (2002), Nr. 1, S. 111–118
- [Pfe08] PFETSCH, Marc E.: Branch-and-Cut for the Maximum Feasible Subsystem Problem. In: *SIAM J. Optim.* 19 (2008), Nr. 1, S. 21–38
- [PR96] PARKER, Mark ; RYAN, Jennifer: Finding the minimum weight IIS cover of an infeasible system of linear inequalities. In: *Ann. Math. Artif. Intell.* 17 (1996), Nr. 1-2, S. 107–126
- [Sch00] SCHMIDT, Karsten: LoLA: A Low Level Analyser. In: MOGENS NIELSEN AND DAN SIMPSON (Hrsg.): *Application and Theory of Petri Nets, 21st International Conference (ICATPN 2000)*, Springer-Verlag, Juni 2000 (Lecture Notes in Computer Science 1825). – ISBN 3–540–67693–7, S. 465–474
- [Sch04] SCHMIDT, Karsten: *Automated generation of a progress measure for the sweep-line method.* Jensen, Kurt (Hrsg.) et al., Tools and algorithms for the construction and analysis of systems. 10th international conference, TACAS 2004, held as part of the joint conferences on theory and practice of software, ETAPS 2004, Barcelona, Spain, March 29 – April 2, 2004. Proceedings. Berlin: Springer. Lecture Notes in Computer Science 2988, S. 192-204, 2004
- [Val90] VALMARI, Antti: A Stubborn Attack On State Explosion. In: CLARKE, Edmund M. (Hrsg.) ; KURSHAN, Robert P. (Hrsg.): *CAV Bd. 531*, Springer, 1990 (Lecture Notes in Computer Science). – ISBN 3–540–54477–1, S. 156–165