Ein effizientes Optimierungsverfahren für Petri-Netz-Modelle von Fertigungssystemen

Armin Zimmermann
Institut für Technische Informatik
Technische Universität Berlin
Franklinstr. 28/29, Sekr. FR 2-2
10587 Berlin
Tel.: +49 (30) 314 - 73 112
Fax: +49 (30) 314 - 21 116
E-Mail: azi@cs.tu-berlin.de

Diego Rodriguez, Manuel Silva
Informatik und Systemtechnik
Universität Zaragoza
c/Maria de Luna 3
Zaragoza 50015 Spanien
Tel.: +34 (976) 76 - 2337
Fax: +34 (976) 76 - 2111
E-Mail: {dierodri, silva}@posta.unizar.es

Abstract: Die Planung von Fertigungssystemen kann mit modellbasierter Leistungsbewertung sicherer und kostengünstiger durchgeführt werden. Mit Hilfe von quantitativen Bewertungsverfahren ist eine automatische Optimierung möglich, die jedoch häufig sehr rechenzeitintensiv ist. Dieser Beitrag beschreibt eine Methode zur automatischen Optimierung von Fertigungssystemen, die für bisher betrachtete Beispiele eine Beschleunigung um den Faktor 35 gegenüber normalem *simulated annealing* ermöglicht. Dies wird durch eine Voroptimierungsphase erreicht, die erfolgversprechende Bereiche des Parameterraumes mit einer schnellen Approximationsmethode bestimmt. Möglich ist dies durch die Modellierung der Fertigungssysteme mit Petri-Netzen. Schätzungen der Leistungskennzahlen werden aus ihren oberen und unteren Schranken berechnet, die sich sehr effizient direkt aus der Netzstruktur ableiten lassen.

Stichworte: Optimierung, Fertigungssysteme, Petri-Netze, Leistungsbewertung

1 Einleitung

Die Planung von modernen Fertigungssystemen ist eine komplexe Aufgabe. Eine wichtige Frage ist dabei, ob das geplante System später den gestellten Anforderungen an den Durchsatz o.ä. gerecht wird. Für die Vorhersage solcher Größen kann eine modellbasierte Leistungsbewertung wertvolle Unterstützung liefern. Die reine Bewertung einer bestimmten Auslegung ist aber oft nicht genug. Häufig sind verschiedene Möglichkeiten für eine Reihe von Entscheidungen vorhanden, von denen allein aufgrund ihrer Anzahl nicht jede einzelne Kombination untersucht werden kann. Hier setzt die automatische Optimierung an, die mit Hilfe modellbasierter Leistungsbewertung aus einer Reihe von Möglichkeiten die optimale berechnet.

Für die Modellierung von Fertigungssystemen werden in der vorliegenden Arbeit stochastische zeiterweiterte Petri-Netze [Rei85, ABCDF95] eingesetzt. Diese Art des Beschreibungsmittels hat sich für die Modellierung von Systemen bewährt, in denen konkurrierende, synchrone, verteilte und parallele Vorgänge auftreten. Fertigungssysteme sind eines der klassischen Anwendungsgebiete [SilTer97]. Ihre grafische Darstellungsform sowie die umfangreichen Arbeiten zu verschiedenen Analyseverfahren sind für ihre Anwendung vorteilhaft.

Bei der Auswahl zwischen verschiedenen Möglichkeiten der Auslegung eines Fertigungssystems treten verschiedene Typen von Entscheidungen auf. Die Anzahl von gleichartigen Maschinen, Werkstückträgern, die Puffergröße und ähnliche Fragen entsprechen der Auswahl einer natürlichen Zahl. Eine reelle Zahl wird beispielsweise bei der Entscheidung über einen Produktmix gesucht. Darüber hinaus gibt es reine Auswahlentscheidungen, bei denen nicht einmal mehr eine sinnvolle Reihenfolge zwischen den möglichen Werten angegeben werden kann. Beispiele dafür sind die Auswahl einer bestimmten Maschine, die Auslegung entsprechend einer Transportstrategie, die Aufteilung der Mitarbeiter und ähnliches. Üblicherweise treten die genannten Probleme in praktischen Optimierungsproblemen gemischt auf. Die genannten Eigenschaften machen Fertigungssysteme für Standardverfahren der Optimierung nur mit Anpassungen zugänglich, da diese häufig nur auf reellen Zahlen arbeiten.

Zur Bewertung, welche Variante wie gut ist, wird neben dem Modell des Fertigungssystems eine Optimierungsfunktion (oft auch Kostenfunktion genannt) benötigt. Diese muß alle für die Bewertung bedeutsamen Faktoren erfassen, beispielsweise alle Kosten und Erlöse einer Anlage für eine ausgewählte Zeiteinheit. Dabei können Werte weggelassen werden, die nicht durch eine Änderung der Parameter beeinflußt werden. Für Fertigungssysteme sind die sich ergebenden Funktionen häufig nichtlinear in einem hochdimensionalen Parameterraum, was die Anwendung einfacher Optimierungsverfahren verhindert.

Für die Durchführung der Optimierung stehen zahlreiche Algorithmen zur Verfügung. Generative Verfahren berechnen aus einer Menge von Einschränkungen und Bedingungen direkt die optimale Lösung. Beispiele dafür sind Probleme der linearen oder ganzzahligen Programmierung. Sie sind aber für die Optimierung selbst einfacher Fertigungssysteme nicht anwendbar, da diese ein zu komplexes Verhalten haben. Statt dessen müssen iterative Verfahren eingesetzt werden, die in vielen Schritten jeweils für einen Parametersatz die Optimierungsfunktion berechnen lassen und deren Optimum suchen.

Die Lösung nichttrivialer Optimierungsprobleme von Fertigungssystemen ist normalerweise NP-hart, so daß nur das Untersuchen des gesamten Parameterraumes eine Garantie für das Finden des tatsächlichen Optimums bietet. Leider ist dies technisch kaum möglich, da die Anzahl der Parameterkombinationen zu groß ist. Für derartige Probleme existieren verschiedene heuristische Suchverfahren, die nur einen geringen Teil der Parameterkombinationen untersuchen, aber trotzdem akzeptable Lösungen nahe des Optimums finden. Dazu gehören Verfahren wie *tabu search* [Ree93], *genetic algorithms* [Ree93, GenCh97], oder *simulated annealing* [AarKor89, Ing96]. Letzteres wird in dieser Arbeit verwendet und in Abschnitt 4.1 näher erläutert.

Zur Bestimmung des Wertes der Optimierungsfunktion für einen bestimmten Parametersatz können die für Petri-Netze bekannten Verfahren verwendet werden. Dazu gehören exakte und approximative analytische Verfahren, Simulation oder netzbasierte Approximationsverfahren. Ein Überblick über verschiedene Methoden gibt [BalSil98]. Für die genaue Bewertung wird in dieser Arbeit Simulation verwendet, da diese für Modelle ohne die Einschränkungen der numerischen Analyse beispielsweise in Bezug auf die Größe des Erreichbarkeitsgraphen anwendbar ist.

In einer vorangegangenen Arbeit [ZimRodSil99] wurde festgestellt, daß die Anwendung von Heuristiken wie *simulated annealing* die Optimierung zwar beschleunigen, sie aber aufgrund der vielen notwendigen Simulationsläufe trotzdem sehr rechenintensiv bleibt. Daraus entstand die Idee, die Optimierung in zwei Phasen zu teilen. In der ersten Voroptimierungsphase werden keine Simulationen durchgeführt, sondern statt dessen approximative, schnelle Berechnungen der Optimierungsfunktion. Dazu werden Algorithmen verwandt, die die Bestimmung oberer und unterer Schranken für quantitative Kennzahlen eines Petri-Netzes direkt aus der Modellstruktur ermöglichen. Nach dem Ende der ersten Phase liegt als Ergebnis der Parametersatz vor, für den die approximierte Funktion optimal ist (zumindest soweit, wie der zugrunde liegende Optimierungsalgorithmus das Optimum gefunden hat).

An dieser Stelle im Parameterraum beginnt nun die Suche nach dem tatsächlichen Optimum in der zweiten Phase. In dieser wird beispielsweise Simulation verwendet, um genaue Ergebnisse für die Optimierungsfunktion zu berechnen. Da der Ausgangspunkt wahrscheinlich bereits in der Nähe des Optimums liegt, kann diese zweite Phase durch Verkürzung der Suche beschleunigt werden. Darüber hinaus wird eine Art *cache* für bereits berechnete Ergebnisse eingesetzt, da sonst der Optimierungsalgorithmus häufig ähnliche oder gleiche Parametersätze mehrfach untersucht.

Im Anschluß wird ein Anwendungsbeispiel dargestellt, an dem später die Methode erläutert wird. Nach der Erläuterung der Approximationsmethode und dem vorgeschlagenen Optimierungsverfahren wird außerdem kurz eine prototypische Implementation der Methode vorgestellt.

2 Ein Anwendungsbeispiel

Zur Erläuterung der Optimierungsmethode wird ein Modell einer Montagelinie verwendet. Drei verschiedene Werkstücke, im folgenden mit A, B und C bezeichnet, werden von fünf Montagestationen zu einem Endprodukt montiert. Einige der Stationen werden für mehrere Montageschritte benötigt, so daß eine gemeinsame Nutzung derselben Betriebsmittel notwendig wird. Die Montage läuft wie folgt ab: A-Teile werden bei Station 1 eingeschleust, an der ein weiteres, hier nicht näher betrachtetes Teil montiert wird. Anschließend werden sie zu Montagestation 3 transportiert, wo ein weiteres zusätzliches Teil angebaut wird. Werkstücke vom Typ B werden an Station 2 eingelagert. Nach einer Montageoperation mit einem unspezifizierten Teil in dieser und der folgenden Station 1 werden sie mit jeweils einem vormontierten A-Teil in Montagestation 4 zusammengefügt. Teile vom Typ C werden in den Stationen 3, 1 und 4 vormontiert, bevor sie an Station 5 mit dem aus A und B gefertigten Werkstück zum Endprodukt montiert werden.

Die betrachtete Aufgabe des Montagesystems ist die Erfüllung von Kundenbestellungen. Es wird angenommen, daß nur bestellte Endprodukte verkauft werden, und daß die Wartezeit eines Kunden zwischen Bestellung und Lieferung Einfluß auf den zu erzielenden Preis hat. Die Anlage wird daher nicht mit dem maximal möglichen Durchsatz gefahren, der natürlich größer sein muß als die zu erwartenden Bestellungen pro Zeiteinheit. Um eine kleinstmögliche Wartezeit zu erzielen, sollten Produkte auf Vorrat produziert und im Lager gehalten werden, um sofort auf Bestellungen reagieren zu können. Der Nachteil dieser Vorgehensweise sind jedoch die mit der Lagerhaltung verbundenen Kosten. Für das Anwendungsbeispiel werden daher verschiedene Produktionsalternativen untersucht, um die richtige Balance zwischen geringen Wartezeiten und Lagerhaltungskosten zu erreichen. Drei verschiedene Strategien werden dabei bewertet, die hier als "Lager", "Bestellung" und "Kanban" bezeichnet werden.

2.1 Modellierung des Beispiels

Im folgenden werden die Modelle der drei Auslegungen der Montagelinie erläutert. Als Beschreibungsmittel werden stochastische zeiterweiterte Petri-Netze verwendet (generalized stochastic Petri nets [ABCDF95]). Dies vereinfacht die Darstellung der Optimierungsmethode, obwohl die im weiteren erläuterten Verfahren grundsätzlich auch für farbige Petri-Netze [Jen92] anwendbar sind. Für komplexere Fertigungssysteme bieten diese bessere Beschreibungsmöglichkeiten [ZimHom99].

Bild 1 zeigt das Montagesystem mit der Strategie "Lager", bei der Teile ohne auf Bestellungen zu warten produziert und im Lager gehalten werden (*push*). Die drei Ausgangsteile A, B und C werden im Modell in den Stellen `inA`, `inB` und `inC` angeliefert. Die anfängliche Anzahl der Teile N wird so gewählt, daß alle Zwischenpuffer für jeden Wert von B (siehe weiter unten) gefüllt werden können. Montagestationen werden durch eine Resourcenstelle mit dem Namen der Station dargestellt (z.B. `M1`), einer Abfolge zeitloser Transitionen, einer Bearbeitungsstelle (wie `m1A`), und der eigentlichen Bearbeitungstransition (z.B. `M1A`) modelliert. Außer den Montagestationen sind neun Zwischenpuffer vorhanden (`B1...B9`), die jeweils eine Kapazität von B Werkstücken haben. B ist einer der veränderlichen Parameter der Optimierung und beeinflußt die maximale Anzahl der Teile in Bearbeitung (Durchlaufbestand). Ausfälle von Maschinen sowie Transportvorgänge werden in diesem Modell nicht betrachtet.

Die Kunden werden im oberen rechten Teil des Bildes modelliert. Es gibt U Kunden, die entweder aus Modellsicht inaktiv sind (eine Marke ist in Stelle `cust`) oder auf die Auslieferung der bestellten Produkte warten (Marke in Stelle `wait`). Die mittlere Schaltzeit der Transition `Dem` entspricht der erwarteten Zeit zwischen zwei Bestellungen eines Produktes. Fertige Produkte werden im Ausgangspuffer `B8ABC`) abgelegt, von wo sie durch das Schalten der Transition `ok` auf eine vorliegende Bestellung hin ausgeliefert werden.

Bild 2 zeigt dieselbe Montagelinie unter Berücksichtigung der Produktion ausschließlich auf Bestellung (*pull*). Jede Bestellung (Transition `Dem` schaltet) startet die Montage eines neuen Fertigprodukts, indem die Ausgangsteile in den Stellen `inA`, `inB` und `inC` abgelegt werden. Solange keine Bestellungen vorliegen, ist das Montagesystem leer und inaktiv. Das Schalten der Transition `M5ABC` steht für die letzte Montageoperation sowie das Ausliefern, nach dem der bestellende Kunde wieder in den inaktiven Zustand übergeht. Auch in diesem Modell kann der Durchlaufbestand durch den Parameter B beeinflußt werden.

Das Modell der Montagelinie mit Kanban-Strategie ist in Bild 3 dargestellt. Im gewählten Beispiel wurden die Stationen in zwei Kanban-Sektoren eingeteilt, die durch eine zentrale zeitlose Transition verbunden sind. Die Anzahl der Kanban-Karten der beiden Sektoren werden mit $K1$ und $K2$ festgelegt. Jeder Sektor produziert Teile und lagert sie in seinem Ausgangspuffer, solange noch freie Kanban-Karten vorhanden sind. Die Anzahl der Karten steuert daher direkt den Durchlaufbestand jedes Sektors und wird während der Optimierung festgelegt.

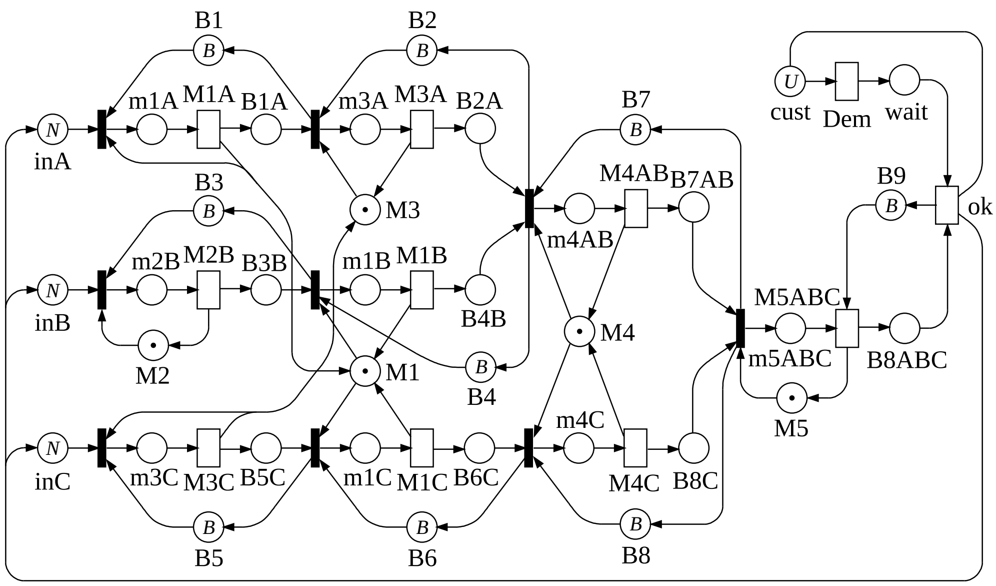

Abbildung 1: Montagelinie mit fünf Stationen; Lager-Strategie

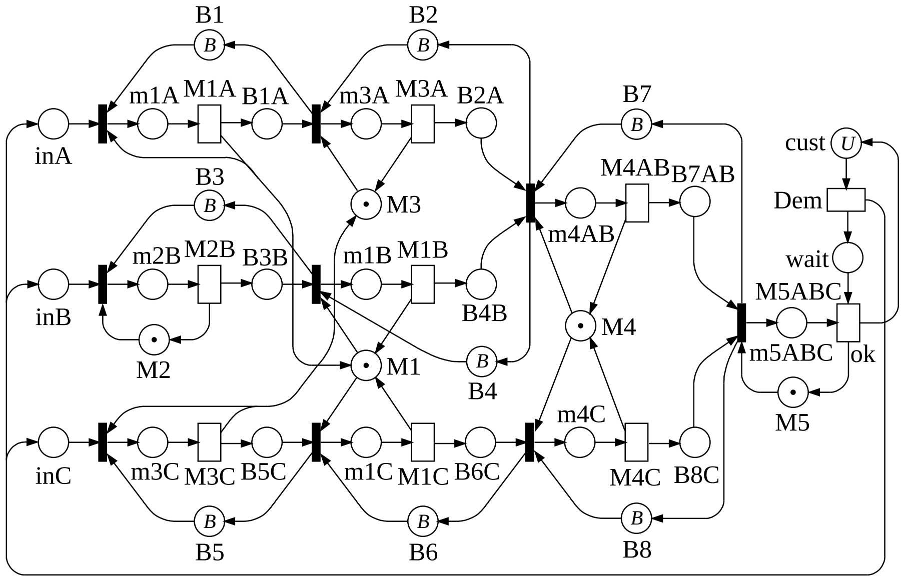

Abbildung 2: Montagelinie mit Bestell-Strategie

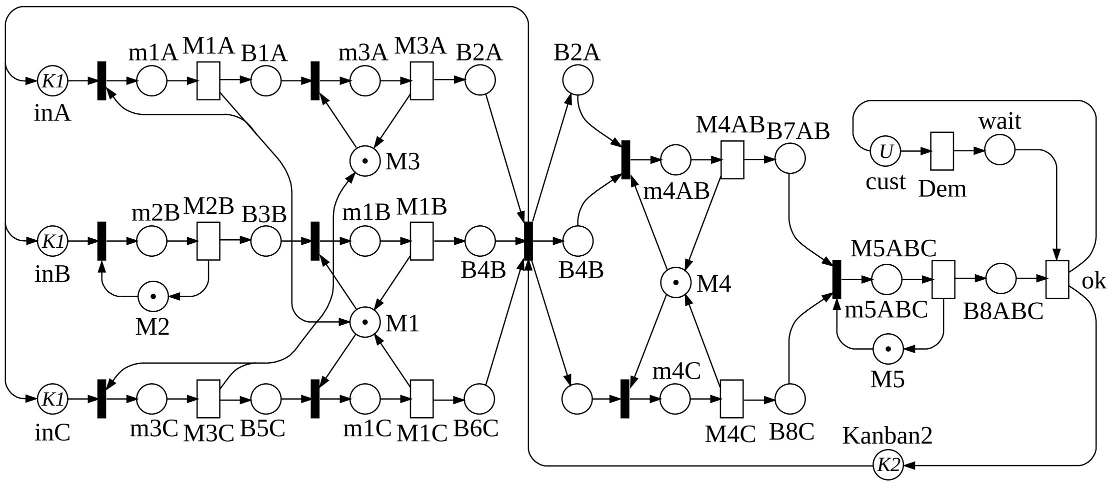

Abbildung 3: Montagelinie mit Kanban-Strategie

2.2 Parameter des Optimierungsproblems

Das Beispiel und die dazugehörigen Optimierungsparameter wurden gewählt, um möglichst verschiedene Typen von Parametern mit der dargestellten Methode untersuchen zu können. Die folgenden Parameter werden durch die Optimierung bestimmt:

- Eine der bei der Modellbeschreibung dargestellten Materialflußstrategien muß ausgewählt werden:
  - Die erste Strategie ("Lager"oder *push*) minimiert die Wartezeit auf Kosten hoher Durchlaufbestände.
  - Im Gegensatz dazu wird bei der Produktion auf Bestellung (*pull*) der Durchlaufbestand auf Kosten der Wartezeit minimiert.

  - Eine dazwischen liegende Lösung ist die Kanban-Strategie, bei der der Durchlaufbestand direkt mit Hilfe von Kanban-Karten beeinflußt werden kann.

- Ein weiterer wichtiger Parameter ist die Größe der Zwischenlager (bzw. Anzahl der Kanban-Karten) im Montagesystem. Diese werden durch Festlegung der Parameter B, $K1$ und $K2$ beeinflußt.

- Außerdem wird ein Auswahlproblem zwischen verschiedenen Montagestationen betrachtet. Es wird angenommen, daß es für die Beschaffung der Station 1 drei verschiedene Alternativangebote gibt. Diese unterscheiden sich in ihrer Geschwindigkeit, wobei schnellere Stationen teurer sind. Die Geschwindigkeit der Maschine wird im Modell durch die Schaltzeit $M1Delay$ der Transitionen `M1A`, `M1B` und `M1C` eingestellt.

Für die Bestimmung einer "optimalen" Auslegung der Montagelinie unter Berücksichtigung der veränderbaren Parameter muß nun noch eine Optimierungsfunktion festgelegt werden. Diese mißt den fiktiven Gewinn der Montagelinie pro gewählter Zeiteinheit.

In die Berechnung des Erlöses durch Verkauf der Fertigprodukte wird die Annahme einbezogen, daß Kunden um so weniger für ein Produkt bezahlen, je länger sie nach der Bestellung auf die Lieferung warten mußten.

Der Durchlaufbestand führt zu Kapitalbindungskosten. Er wird im Modell aus den mittleren Lagerfüllständen und den weiteren möglichen Orten der Werkstücke bestimmt. Die Größe der Zwischenlager sowie die Auswahl der Station 1 beeinflussen die Kosten für Investment bzw. Amortisation, Betrieb und Wartung. Außerdem wird ein konstanter Betrag für weitere Fixkosten angenommen.

3 Approximation von Leistungskennzahlen

Für eine Optimierung in zwei Phasen, wie in dieser Arbeit vorgeschlagen, wird für die Voroptimierung eine schnelle approximative Methode zur Berechnung quantitativer Leistungsgrößen benötigt. Hierfür ist das Beschreibungsmittel Petri-Netze vorteilhaft, da bereits in vielen Arbeiten unterschiedliche Methoden dafür entwickelt wurden.

3.1 Berechnung von Schranken für Leistungsgrößen

Die im folgenden vorgestellte Methode benötigt ausschließlich Informationen aus der Modellstruktur und kommt ohne eine Betrachtung der Zustände des Systems aus. Sie ist daher um Größenordnungen schneller als etwa numerische Analyse oder Simulation, dies wird aber mit einer geringen Genauigkeit erkauft. Sie verwendet die durch *Campos* und andere vorgestellten Verfahren der Berechnung oberer und unterer Schranken [BalSil98, CamSil92, Cam90] für Durchsätze von Transitionen und mittlere Markenanzahlen in Stellen. Der Durchsatz entspricht dabei der mittleren Schaltanzahl pro Zeiteinheit.

Diese Methode ist für stochastische Petri-Netze im allgemeinen anwendbar, jedoch für den hier eingesetzten Spezialfall der sogenannten FRT-Netze (*freely related t-invariants*) einfacher zu realisieren. Modelle dieser Netzklasse sind vollständig durch Transitions-Invarianten bedeckt, deren Durchsätze in einem festen Verhältnis untereinander stehen. Für zahlreiche Modelle von Fertigungssystemen stellt dies keine Einschränkung dar.

Für diese Netzklasse können *routing*-Gleichungen aufgestellt werden [CamSil92]. Für jede Untermenge von Transitionen $T_i = \{t_1, t_2, ..., t_k\} \subset T$ aller Transitionen T, die untereinander in gleichem Konflikt (*equal conflict relation*) stehen (d.h. ihre Eingangskanten sind gleich: $\mathbf{Pre}[\cdot, t_1] = ... = \mathbf{Pre}[\cdot, t_k]$) gelten folgende Gleichungen:

$$\begin{aligned} r_2\mathbf{v}[t_1] - r_1\mathbf{v}[t_2] &= 0 \\ r_3\mathbf{v}[t_2] - r_2\mathbf{v}[t_3] &= 0 \\ &\ldots \\ r_k\mathbf{v}[t_{k-1}] - r_{k-1}\mathbf{v}[t_k] &= 0 \end{aligned} \tag{1}$$

wobei $\mathbf{v}[t_k]$ die Besuchsrate (*visit ratio*) der Transition t_k angibt und r_k die relative Besuchsrate im Konflikt der Transition t_k. Letztere entsprechen für zeitlose Transitionen ihren im Modell angegebenen relativen Schaltwahrscheinlichkeiten. Dieses Gleichungssystem kann in Matrixform als $\mathbf{R}[T_i] \cdot \mathbf{v} = 0$ dargestellt und zur sogenannten *routing*-Matrix $\mathbf{R}$ kombiniert werden:

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}[T_1] \\ \vdots \\ \mathbf{R}[T_r] \end{pmatrix} \tag{2}$$

Die Inzidenzmatrix $\mathbf{C} = \mathbf{Pre} - \mathbf{Post}$ wird nun mit $\mathbf{R}$ erweitert, so daß sich das folgende lineare Gleichungssystem ergibt:

$$\begin{pmatrix} \mathbf{C} \\ \mathbf{R} \end{pmatrix} \cdot \mathbf{v}^{(1)} = 0, \ \mathbf{v}^{(1)}[t_1] = 1 \tag{3}$$

Die Lösung dieses Gleichungssystems ergibt die Besuchsraten aller Transitionen des Modells in Relation zu Transition t_1 [BalSil98, CamSil92]. Der Wert von $\mathbf{v}^{(1)}[t_i]$ beschreibt die Schaltanzahl der Transition t_i im eingeschwungenen Zustand für jedes Schalten von t_1.

Unter Verwendung dieser Ergebnisse können die mittleren *service demands* $\overline{\mathbf{D}}^{(1)}$ jeder Transition in Relation zu t_1 berechnet werden durch:

$$\overline{\mathbf{D}}^{(1)}[t_i] = v^{(1)}[t_i] \cdot S[t_i] \tag{4}$$

wobei $v^{(1)}[t_i]$ die Besuchsrate von Transition t_i relativ zu t_1 ist und $S[t_i]$ die mittlere Schaltdauer von Transition t_i beschreibt (im Modell gegeben).

Nun kann eine obere Schranke $\chi_+[t_1]$ für den Durchsatz von Transition t_1 aus dem Kehrwert des Ergebnisses des folgenden Problems der linearen Programmierung (LPP) berechnet werden [CamSil92]:

$$\begin{array}{rlrcl}
\chi_+[t_1] = \frac{1}{h_1} & & & & \\
\text{mit } h_1 = & \text{maximiere} & y \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}}^{(1)} & & \\
& \text{für} & y \cdot \mathbf{C} & = & 0 \\
& & y \cdot M_0 & = & 1 \\
& & y & \geq & 0
\end{array} \tag{5}$$

wobei y für Stellen-Invarianten steht. Im nächsten Schritt können obere Schranken für den Durchsatz aller Transitionen berechnet werden, indem die Schranke für t_1 mit der Besuchsrate der jeweiligen Transition multipliziert wird.

Im Anschluß werden untere Schranken für die Durchsätze der Transitionen berechnet [CamSil92]:

$$\begin{array}{rcl}
\chi_-[t_1] & = & \frac{1}{\sum_{t \in T} \overline{\mathbf{D}}^1[t]} \\
\chi_-[t_i] & = & \chi_-[t_1] \cdot \mathbf{v}^{(1)}[t_i]
\end{array} \tag{6}$$

Eine untere Schranke für die mittlere Markenanzahl M_- in den Stellen kann unter Verwendung der unteren Schranke für den Durchsatz für alle Stellen gemeinsam erfolgen [CamSil92]:

$$M_- = \mathbf{Pre} \cdot \overline{\mathbf{D}}^{(1)} \cdot \chi_-^{(1)}[t_1] \tag{7}$$

Zur Bestimmung einer oberen Schranke der mittleren Markenanzahl muß folgendes LPP für jede Stelle p_i gelöst werden:

$$\begin{array}{rlrcl}
M_+[p_i] = & \text{minimiere} & M_-[p_i] + Y^T \cdot (M_0 - M_-) & & \\
& \text{für} & Y^T \cdot \mathbf{C} & = & 0 \\
& & Y^T \cdot \mathbf{e}_i & = & 1 \\
& & Y & \geq & 0
\end{array} \tag{8}$$

mit

$$\mathbf{e}_i[k] = \begin{cases} 1 & \text{falls } i = k \\ 0 & \text{sonst} \end{cases} \tag{9}$$

Die Lösung der genannten Probleme der linearen Programmierung kann beispielsweise mit Hilfe des Simplex-Algorithmus sehr effizient erfolgen.

Für den Spezialfall von Transitionen t_r mit nur einer Eingangsstelle p_i kann folgende Formel als zusätzliche Einschränkung in einigen Fällen die Schätzung verbessern:

$$M_+[p_i] \leq K \cdot \chi_+[t_r] \cdot S[t_r] + (K - 1) \tag{10}$$

wobei $K = \mathbf{Pre}(p_i, t_r)$ und $S[t_r]$ die Schaltzeit der Transition t_i angibt.

3.2 Heuristiken zur Approximation der Leistungsgrößen

Ein approximierter Wert $\chi_{\simeq}[t_i]$ für den Durchsatz einer Transition t_i wird aus folgender gewichteter Summe der oberen und unteren Schranke berechnet:

$$\chi_{\simeq}[t_i] = \alpha \cdot \chi_{+}[t_i] + (1-\alpha) \cdot \chi_{-}[t_i] \tag{11}$$

Die bisherigen Erfahrungen mit zahlreichen Modellen zeigte, daß der tatsächliche Wert sehr viel näher zur oberen Schranke liegt, die darüber hinaus stärker auf Änderungen von Parametern reagiert. Dies ist auch im Hinblick auf die vergleichsweise einfache Berechnungsart der unteren Schranke zu erwarten. Für den Wert α wurde daher 0.9 gewählt.

Ähnlich wird die mittlere Markenanzahl $M_{\simeq}[p_i]$ in einer Stelle p_i berechnet:

$$M_{\simeq}[p_i] = \beta \cdot M_{+}[p_i] + (1-\beta) \cdot M_{-}[p_i] \tag{12}$$

β ist hier der Gewichtungsfaktor, der aufgrund bisheriger Erfahrungen auf den Wert 0.5 festgelegt wurde. Die Schranken der Markenanzahl liegen oft nicht sehr eng beieinander, aber bisher konnte keine systematische Richtung festgestellt werden.

Wie bereits gesagt sind die unteren Schranken für Durchsätze von Transitionen nicht so genau wie die oberen. Die Schranken für die Markenanzahlen werden auf Grundlage dieses Wertes berechnet und liegen daher häufig weit auseinander. Da in diesem Fall keine exakte Schranke benötigt wird, sondern eine möglichst gute Approximation des tatsächlichen Wertes, wird folgende Änderung verwendet.

Es wird nach der Durchsatzschätzung angenommen, daß $\chi_{\simeq}$ (berechnet nach Gleichung (11)) dem tatsächlichen Wert des Durchsatzes entspricht. Die oberen und unteren Schranken des Durchsatzes in den Gleichungen (10) und (7) können dann durch $\chi_{\simeq}$ ersetzt werden.

Nach dieser Änderung sind die Schranken der Markenanzahlen nur noch approximierte Werte, die nicht unbedingt den tatsächlichen Wert einschließen. Der geschätzte Wert der Markenanzahl kann jedoch in einigen Fällen signifikant verbessert werden.

3.3 Approximative Untersuchung des Beispiels

Die Werte der Optimierungsfunktion werden nun für ausgewählte Bereiche des Parameterraumes für das Anwendungsbeispiel berechnet. Bild 4 zeigt die Ergebnisse für die Strategie „Lager“ mit variierender Auswahl der Station 1 und Zwischenlagergröße. Die Anwendung der geänderten Schätzung für die Markenanzahlen verbessert das Ergebnis. Entscheidend für eine Bewertung der Qualität der Approximation ist nicht so sehr der absolute Wert der Optimierungsfunktion, als vielmehr eine Übereinstimmung in der Form der Kurven. Liegen Optimimum der approximierten und tatsächlichen Funktion an derselben Parameterstelle, ist unabhängig vom geschätzten Wert der Optimierungsfunktion das Ergebnis der ersten Optimierungsphase bereits nicht mehr zu verbessern.

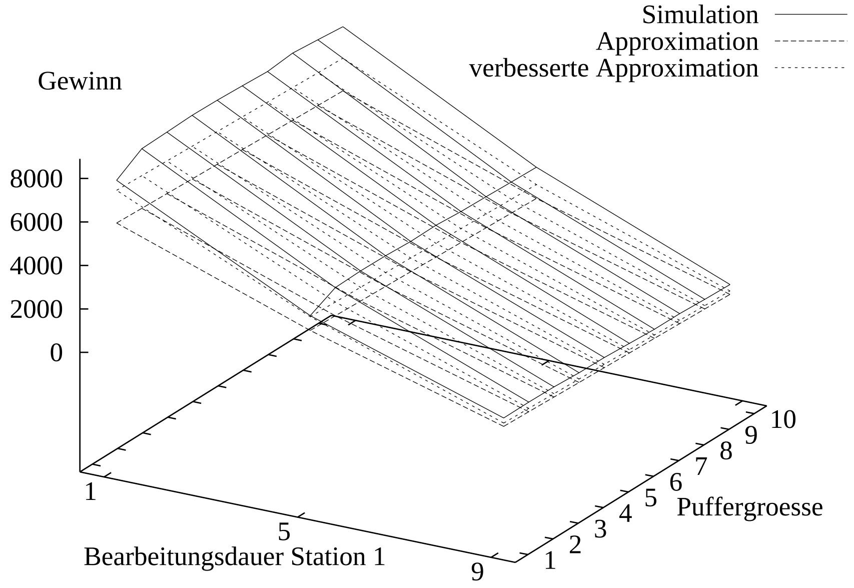

Abbildung 4: Optimierungsfunktion für Lager-Strategie

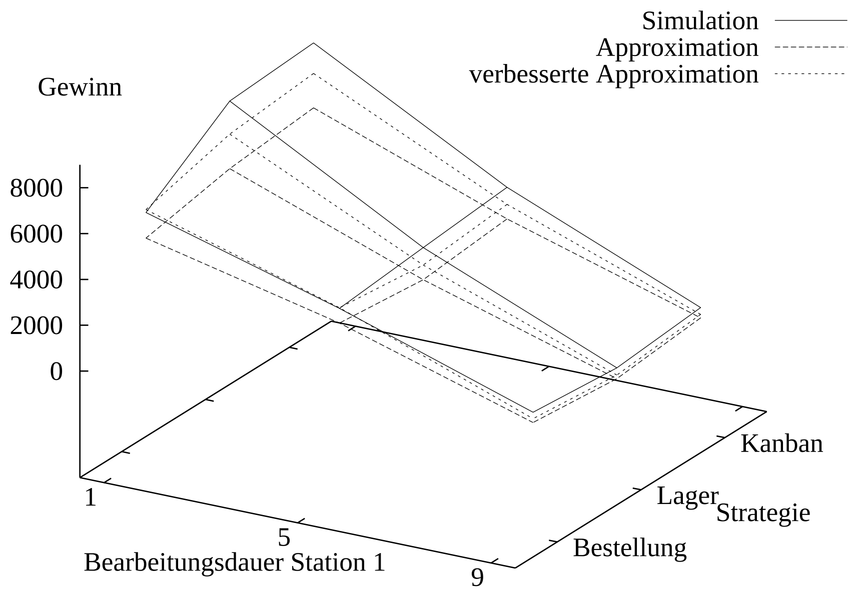

Abbildung 5: Optimierungsfunktion für verschiedene Strategien

Bild 5 zeigt einen anderen Ausschnitt der Optimierungsfunktion, hier in Abhängigkeit von gewählter Strategie sowie Station 1. Wiederum erfassen die approximierten Werte die tatsächliche Kurvenform sehr gut.

4 Ein effizientes Optimierungsverfahren

Das in dieser Arbeit vorgeschlagene Optimierungsverfahren nimmt zwei Änderungen an einem Standardverfahren (siehe unten) vor:

Bei der Verwendung des normalen Optimierungsverfahrens fiel auf, daß die Optimierungsfunktion häufig für ähnliche oder gleiche Parametersätze aufgerufen wurde. Um diesen unnötigen Aufwand zu sparen, wird ein Ergebnisspeicher (*cache*) eingesetzt. Eine aufwendige Leistungsuntersuchung z.B. mit Simulation wird dann nur noch vorgenommen, wenn der Parametersatz sich nicht innerhalb festgelegter Grenzen um einen bereits untersuchten befindet. Ansonsten kann einfach auf das bereits vorliegende Ergebnis zurückgegriffen werden.

Die Optimierung wird in zwei aufeinander aufbauende Phasen unterteilt. Die erste Voroptimierungsphase besteht aus dem normalen Optimierungsalgorithmus, der jedoch für jede Auswertung der Optimierungsfunktion für einen bestimmten Parametersatz keine aufwendige Analyse startet, sondern die in Abschnitt 3.2 beschriebenen effizienten Approximationsverfahren. Dadurch wird das Optimum der approximierten Funktion gefunden.

Der Parametersatz dieses Optimums wird nun als Startwert (Ausgangspunkt der Suche) für die zweite Phase verwendet. Dabei wird das normale Bewertungsverfahren zur Berechnung der Optimierungsfunktion benutzt, in diesem Fall Simulation. Da der Startpunkt mit einer gewissen Wahrscheinlichkeit in der Nähe des tatsächlichen Optimums liegt, kann die Suche beschleunigt werden. Dazu muß der Optimierungsalgorithmus geeignet angepaßt werden (siehe folgender Abschnitt für das gewählte Beispiel). Als Ausgangsalgorithmus wurde hier *simulated annealing* gewählt, die Methode kann aber auch mit anderen Verfahren kombiniert werden.

4.1 Optimierungsalgorithmus Simulated Annealing

Der in dieser Arbeit verwendete Optimierungsalgorithmus *simulated annealing* wurde einer anpaßbaren, frei verfügbaren Implementation ASA [Ing98, Ing96] entnommen. Er ist für die Optimierung von Fertigungssystemen gut geeignet, bei denen durch eine Mischung verschiedener Parameterarten lokale Optima entstehen können. Der Algorithmus orientiert sich an den physikalischen Vorgängen z.B. bei der Abkühlung und dem Erstarren von kristallinen Stoffen. Er generiert in jedem Schritt einen neuen Parametersatz in der Umgebung des bisherigen Punktes und wertet an dieser Stelle die Optimierungsfunktion aus. Ist sie besser als an der bisherigen Stelle, wird der neue Parametersatz akzeptiert. Falls nicht, kann er trotzdem mit einer gewissen Wahrscheinlichkeit übernommen werden, um lokale Optima verlassen zu können.

Wenn D die Anzahl der Parameter x^i, $i = 1 \ldots D$ bezeichnet, und x die reellwertigen Parameter $x^i = [x^i_{min} \ldots x^i_{max}]$, arbeitet der Algorithmus wie folgt. Dabei wird vereinfachend von nur einem Parameter ausgegangen ($D = 1$). Für $D > 1$ müssen alle Operationen auf x für alle x^i ausgeführt werden.

Anfangsparameter x auswählen
Anfangstemperatur $T_0^{opt} > 0$ der Optimierungsfunktion wählen
Anfangstemperatur $T_0^{par} > 0$ der Parameter wählen
akzeptiert := generiert := 0
$x_{best} := x$; $\text{opt}_{best} := \text{opt\_function}(x)$
repeat
    generiert := generiert + 1
    $T^{par} = T_0^{par} e^{-c\,\text{generiert}^{\frac{1}{D}}}$
    $T^{opt} = T_0^{opt} e^{-c\,\text{akzeptiert}^{\frac{1}{D}}}$
    $x := \text{random}[-1 \ldots 1]$
    $d := x_{max} - x_{min}$
    repeat
        $x := x_{best} + \text{sign}(x)\, T^{par} \left[(1 + \frac{1}{T^{par}})^{|x|} - 1 \right] d$
    until $x_{min} \leq x \leq x_{max}$
    $\text{opt} := \text{opt\_function}(x)$
    $\delta := \text{opt} - \text{opt}_{best}$
    if $\delta < 0 \vee \text{random}[0 \ldots 1] < e^{-\delta / T^{opt}}$
        then $x_{best} := x$; $\text{opt}_{best} := \text{opt}$;
        akzeptiert := akzeptiert + 1
until $T^{par} < \epsilon \vee T^{opt} < \epsilon$
$\Rightarrow x_{best}$ ist die gefundene Lösung

Eine wichtige Rolle spielt die gedachte Temperatur T^{opt} der Optimierungsfunktion, die die Akzeptanzwahrscheinlichkeit schlechterer Lösungen beeinflußt. Ihr Standardwert in ASA ist $T_0^{opt} = 1$. Die Temperatur der Parameter T^{par} bestimmt, wie weit weg vom aktuell akzeptierten Punkt der neue Parametersatz generiert wird. Der Standardwert ist hier ebenfalls $T_0^{par} = 1$. Die Geschwindigkeit des Abkühlens (und damit die Zeit bis zum Beenden des Algorithmus) wird durch die Wahl der Konstanten c bestimmt:

$$c = -ln\text{TRatioScale} * e^{-\frac{ln\text{TAnnealScale}}{D}}$$

mit TRatioScale = 0.00001 und TAnnealScale = 100. Der Wert von TAnnealScale wird später zur Beschleunigung des Algorithmus verändert.

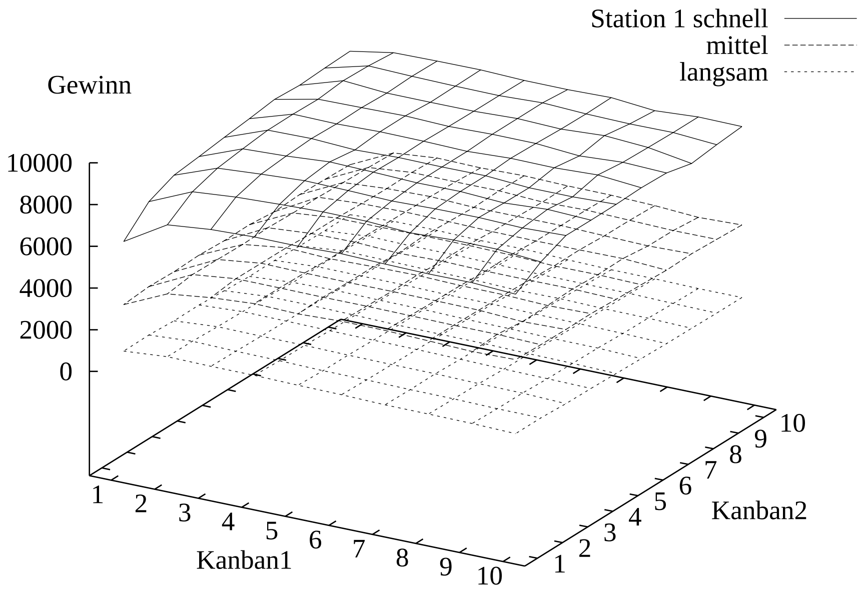

Abbildung 6: Optimierungsfunktion für Kanban-Strategie

4.2 Optimierung des Beispiels

Im folgenden wird die Anwendung des in diesem Beitrag vorgestellten Verfahrens auf das Beispiel aus Abschnitt 2 dargestellt. Alle beschriebenen Berechnungen wurden auf einem Pentium-PC unter Linux durchgeführt. Zur Überprüfung der Güte der durch das Optimierungsverfahren bestimmten Lösung wurde zunächst die Optimierungsfunktion für alle möglichen Parameterkombinationen berechnet. Für reale Probleme wäre dies zu zeitaufwendig. Abb. 6 zeigt das Ergebnis für die Kanban-Strategie, die unter den drei möglichen Strategien die besten Resultate ergab. Darum wurde auf die Darstellung der anderen Ergebnisse verzichtet. Die veränderlichen Parameter sind die Anzahl der Kanban-Karten in den beiden Sektoren, außerdem ist eine Kurve für jede der drei möglichen Stationen 1 gezeigt. Die besten Ergebnisse von etwa 9000 werden für die schnellste Station 1 und Kartenanzahlen von 2 oder größer erzielt.

Zum späteren Vergleich mit der Optimierung in zwei Phasen wurde zunächst eine normale Optimierung mit *simulated annealing* und Ergebnisspeicher gestartet. Jede einzelne Simulation wurde auf 100 Sekunden Rechenzeit beschränkt, um die Gesamtlaufzeit zu begrenzen. Die gesamte Optimierung benötigte 64 Minuten. Als Ergebnis wird ein Gewinn von 8911 für die Kanban-Strategie, $K1 = 4, K2 = 6$ und $M1Delay = 1$ (schnelle Station) gefunden. Dieses in der ersten Spalte der Tab. 1 dargestellte Ergebnis zeigt entsprechend Bild 6, daß optimale Parameter im Rahmen einer sinnvollen Genauigkeit gefunden wurden. In diesem Fall wurden 95 von insgesamt 900 möglichen Parametersätzen untersucht. Dies waren lediglich 19 Prozent der durch den Optimierungsalgorithmus angeforderten Berchnungen, der Rest konnte anhand des

Ergebnisspeichers beantwortet werden. Für Probleme mit größerem Parameterraum benötigt *simulated annealing* üblicherweise einen sehr viel geringeren Prozentsatz von Berechnungen im Verhältnis zur Gesamtanzahl.

Im Anschluß wurde die in dieser Arbeit vorgeschlagene Methode angewandt. Dazu mußte zunächst die Voroptimierung mit der approximativen Bestimmung der Optimierungsfunktion durchgeführt werden. Diese Phase benötigte lediglich 2 Minuten Rechenzeit. 388mal wurde die Optimierungsfunktion aufgerufen, wobei lediglich 59 davon nicht durch Rückgriff auf den Ergebnisspeicher beantwortet werden konnten. Jede einzelne Approximation für einen bestimmten Parametersatz dauerte typischerweise ein bis zwei Sekunden. Für die approximierte Funktion wurde als „Optimum" ein Gewinn von 7761 für Kanban-Strategie, $K1 = 1$, $K2 = 1$ und $M1Delay = 1$ (schnelle Station) gefunden. Der Grund dafür ist darin zu suchen, daß die Approximation nicht den geringen Einfluß der erhöhten Anzahl Kanban-Karten auf den Durchsatz erfaßt, sondern nur die damit verbundenen Kosten. Der tatsächliche Gewinn für diesen Parametersatz beträgt 6115. Die zweite Spalte der Tab. 1 zeigt die Ergebnisse der ersten Phase (Voroptimierung).

	Standard	Phase 1	Phase 2		
TAnnealScale	100	100	10	5	1
Karten K1	4	1	4	4	1
Karten K2	6	1	1	7	1
Station 1	schnell	schnell	schnell	schnell	schnell
Strategie	Kanban	Kanban	Kanban	Kanban	Kanban
Gewinn	8911	6115 (7761)	8702	8649	6115
Zeit (Minuten)	64	2	14	7	2
Beschleunigung			4.0	7.1	16.0

Tabelle 1: Zusammenhang zwischen Beschleunigung und Ergebnisqualität

Die Startwerte der Parameter für die zweite Optimierungsphase werden nun aus dem Ergebnis der Voroptimierung übernommen. Eine Beschleunigung dieser Phase gegenüber einem normalen *simulated annealing* wird durch einen geringeren Wert von TAnnealScale erreicht, der den Abkühlungsfaktor c beeinflußt. Für unterschiedliche Werte wurde nun die zweite Optimierungsphase durchgeführt, um die Grenze der möglichen Beschleunigung aufzuzeigen. Tab. 1 zeigt in den rechten Spalten die Rechenzeiten und Ergebnisse unterschiedlicher Varianten. Unter Inkaufnahme einer geringen Verschlechterung des Ergebnisses kann so eine Beschleunigung um den Faktor 7 erreicht werden, wobei die Dauer der Voroptimierung bereits mit einbezogen wurde. Für TAnnealScale = 1 ist offensichtlich die Grenze des Möglichen überschritten, denn der Optimierungsalgorithmus entfernt sich nicht mehr von der Anfangslösung. Zusammen mit der Beschleunigung durch den Ergebnisspeicher ergibt sich so eine Beschleunigung um den Faktor 35 gegenüber einem normalen *simulated annealing*, das an sich bereits nur einen Bruchteil des Gesamtraums der Lösungen untersucht.

5 Werkzeugunterstützung

Für die Untersuchung der hier vorgestellten Methode an verschiedenen Beispielen wurde ein prototypisches Softwarewerkzeug unter Integration folgender Werkzeuge implementiert:

- Der Optimierungsalgorithmus *simulated annealing* wurde dem frei verfügbaren Programm ASA [Ing98, Ing96] (für *adaptive simulated annealing*) entnommen.
- Zur Erstellung der Petri-Netz-Modelle sowie die Leistungsbewertung mit Simulation wurde das Werkzeug TimeNET [ZFGH00] (timed net evaluation tool) eingesetzt. Die Simulationskomponente dieses Werkzeugs erlaubt die parallele Simulation von stochastischen Petri-Netzen. Während des Simulationslaufs wird fortlaufend die Varianz der Ergebnisse geschätzt, und bei Erreichen der gewünschten Genauigkeit abgebrochen.
- Zur Lösung der für die Approximation notwendigen Probleme der linearen Programmierung wurde das ebenfalls frei verfügbare Paket `lp-solve` eingesetzt.

Abb. 7 zeigt eine Übersicht des Zusammenspiels der verschiedenen Komponenten während einer Optimierung. Voraussetzung sind das Ausgangsmodell sowie Konfigurationsdateien für jede Phase. In ihnen lassen sich die änderbaren Parameter mit Ober- und Untergrenzen sowie Einstellungen der Algorithmen wie z.B. die Temperaturparameter anpassen. Die Definition der Optimierungsfunktion ist Teil des Ausgangsmodells.

Eine neu implementierte Schnittstellenfunktion ersetzt die benutzerdefinierte Optimierungsfunktion von ASA. Diese stellt anhand der Parameterinformationen von ASA und dem Ausgangsmodell ein parametrisiertes Modell zusammen und startet den jeweiligen Auswertungsalgorithmus. Falls der Parametersatz bereits im Ergebnisspeicher vorhanden war, wird das Ergebnis direkt an ASA zurückgegeben. Abhängig von der Optimierungsphase wird zur Auswertung entweder der Simulator von TimeNET oder ein zu diesem Zweck neu implementiertes Programm zur approximativen Bestimmung des Ergebnisses aufgerufen. Letzteres stellt verschiedene Probleme linearer Programmierung in Dateien zusammen, die dann durch `lp-solve` gelöst werden. Das Ergebnis wird von der Schnittstellenfunktion gelesen, im Ergebnisspeicher abgelegt, und anschließend an ASA weitergegeben.

6 Zusammenfassung

Die automatische Optimierung komplexer Fertigungssysteme ist sehr rechenzeitintensiv, selbst wenn Optimierungsstrategien wie beispielsweise *simulated annealing* angewandt werden. Das Problem ist die Anzahl der durchzuführenden Leistungsbewertungen z.B. durch Simulationen. In der vorliegenden Arbeit wird ein Standardoptimierungsverfahren mit Hilfe eines Zwischenspeichers von Ergebnissen sowie einer Voroptimierungsphase beschleunigt. Die Vorphase bestimmt mit Hilfe approximativer, schneller Leistungsbewertung den Startbereich der eigentlichen Optimierung, die von diesem Startpunkt aus sehr viel schneller das Optimum findet.

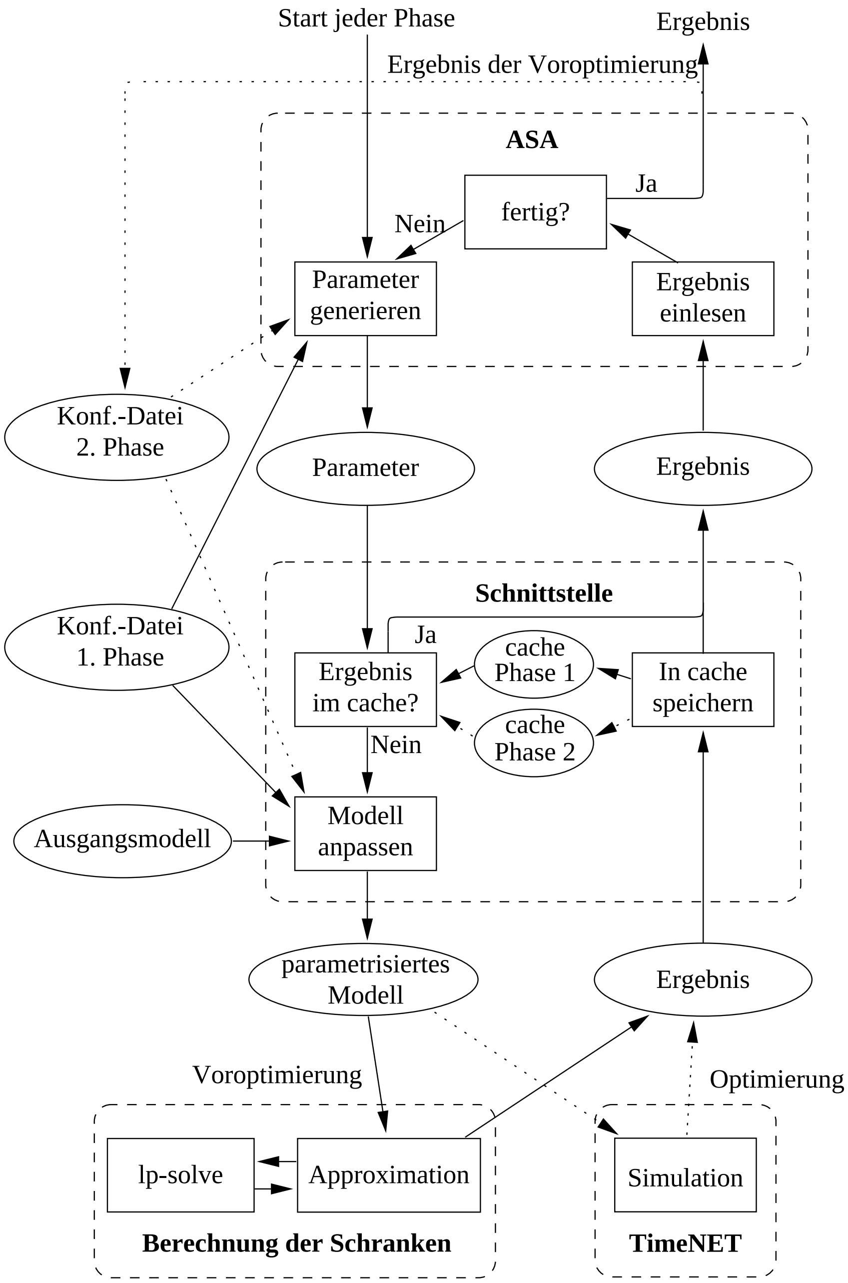

Abbildung 7: Zusammenspiel der Module des Werkzeugs

Insgesamt wird eine typische Beschleunigung um den Faktor 35 für die bisher untersuchten Anwendungsbeispiele erzielt. Die Methode läßt sich mit modernen Optimierungsstrategien kombinieren, so daß insgesamt eine noch sehr viel stärkere Reduktion des Aufwandes im Vergleich zu einer vollständigen Durchsuchung des Parameterraumes möglich ist. Die Arbeit zeigt die Anwendung dieser Methode anhand eines mit stochastischen Petri-Netzen modellierten Fertigungssystems sowie einer prototypischen Implementation des Verfahrens.

Literatur

[AarKor89] E. Aarts und J. Korst, *Simulated Annealing and Bolzmann Machines* (Wiley, 1989).

[ABCDF95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli und G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, Series in parallel computing (John Wiley and Sons, 1995).

[BalSil98] G. Balbo und M. Silva (Eds.), *Performance Models for Discrete Event Systems with Synchronisations: Formalisms and Analysis Techniques* (Universidad de Zaragoza, Spain, 1998), MATCH Advanced School.

[CamSil92] J. Campos und M. Silva, Structural Techniques and Performance Bounds of Stochastic Petri Net models, in: G. Rozenberg, Ed., *Advances in Petri Nets 1992*, *Lecture Notes in Computer Science*, Bd. 609 (Springer Verlag, 1992) 352–391.

[Cam90] J. Campos, *Performance Bounds for Synchronized Queueing Networks*, Ph. D. Dissertation, Universidad de Zaragoza, 1990.

[GenCh97] M. Gen und R. Cheng, *Genetic Algorithms and Engineering Design* (Wiley, 1997).

[Ing98] L. Ingber, Very fast simulated re-annealing, *Journal of Mathematical Computer Modelling* **12** (8) (1989) 967–973.

[Ing96] L. Ingber, Adaptive simulated annealing (ASA): Lessons learned, *Journal of Control and Cybernetics* **25** (1) (1996) 33–54.

[Jen92] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, EATCS Monographs on Theoretical Computer Science (Springer Verlag, 1992).

[Ree93] C. L. Reeves, *Modern Heuristic techniques for Combinatorial Problems* (Wiley, 1993).

[Rei85] W. Reisig, *Petri nets* (Springer Verlag Berlin, 1985).

[SilTer97] M. Silva und E. Teruel, Petri Nets for the Design and Operation of Manufacturing Systems, *European Journal of Control* **3** (3) (1997) 182–199.

[ZimRodSil99] A. Zimmermann, D. Rodriguez und M. Silva, Modelling and Optimisation of Manufacturing Systems: Petri Nets and Simulated Annealing, in: *Proc. European Control Conference (ECC'99)* (Karlsruhe, 1999) .

[ZFGH00] A. Zimmermann, J. Freiheit, R. German und G. Hommel, Petri Net Modelling and Performability Evaluation with TimeNET 3.0, in: *11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation* (Schaumburg, Illinois, USA, 2000) 188–202, LNCS 1786.

[ZimHom99] A. Zimmermann und G. Hommel, Modelling and Evaluation of Manufacturing Systems Using Dedicated Petri Nets, *Int. Journal of Advanced Manufacturing Technology* **15** (1999) 132–137.