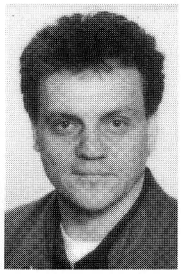
Eine Klasse gerichteter, azyklischer Graphen zur Leistungsbewertung paralleler Systeme

Georg Fleischmann, Kunsthochschule für Medien, Köln,
Matthias Gente, Universität Erlangen-Nürnberg



Prof. Dr. Georg Fleischmann studierte von 1980–86 Informatik in Erlangen. 1986–90 wissenschaftlicher Mitarbeiter am Lehrstuhl für Betriebssysteme der Universität Erlangen und 1990 Promotion zum Dr.-Ing. Seit 1994 Professor für Informatik/ Audiovisuelle Medien an der Kunsthochschule für Medien in Köln. Seine gegenwärtigen Forschungsinteressen umfassen Mensch-Maschine-Kommunikation, interaktive Medien und „Human Figure Animation".

Dipl.-Inf. Matthias Gente studierte von 1983–90 Informatik in Erlangen. Seit 1990 wissenschaftlicher Mitarbeiter am Lehrstuhl für Betriebssysteme der Universität Erlangen-Nürnberg. Sein Forschungsinteresse gilt derzeit der Modellierung und Bewertung von Betriebssystemmechanismen für Parallelrechner.

Die Vorhersage der Laufzeit von parallelen Programmen mit Hilfe von gerichteten, azyklischen Graphen ist bei Verwendung von stochastischen Laufzeiten nur für relativ kleine Probleme möglich. Wir beschreiben eine Klasse von regelmäßigen Graphen, die zur Modellierung und Bewertung von parallelen Systemen besonders geeignet sind. Alle unsere Untersuchungen deuten darauf hin, daß diese regelmäßigen Graphen es erlauben, Ergebnisse von kleinen auf große Graphen mit gleicher Struktur zu extrapolieren. Für einen speziellen Fall regelmäßiger Graphen geben wir eine Beweisskizze an. Durch eine Reihe von Beispielen stützen wir die Annahme.

A Class of Directed Acyclic Graphs for Performance Evaluation of Parallel Systems

The prediction of the completion time of parallel programs by exact evaluation of graph models is only possible for relatively small problems, when stochastic run times are supposed for the nodes. We describe the class of regular graphs which are particularly suited for the modelling and performance evaluation of parallel systems. We assume that regular graphs have a property which allows for the extrapolation of results for small graphs to get good estimates for the total completion time of large graphs. For a special case of regular graphs we give an outline of a proof for that property. We also provide some examples to support our assumption in the general case.

1 Einleitung

Gerichtete, azyklische Graphen eignen sich sehr gut zur Beschreibung von parallelen Algorithmen und werden von vielen Autoren zu diesem Zweck eingesetzt [5; 6; 10; 13], da sie die Struktur eines parallelen Programms in direkter Weise widerspiegeln. Die Knoten dieser Graphen repräsentieren die Teilaufgaben des modellierten parallelen Programms, ihre Kanten beschreiben die Reihenfolgebeziehungen zwischen den Teilaufgaben[1]. Ordnet man den Knoten T_i stochastisch unabhängige Laufzeiten X_i zu, so ergibt sich die Gesamtlaufzeit des Graphen[2] als Maximum über die Laufzeiten aller im Graphen enthaltenen Pfade. Die Pfadlaufzeiten ergeben sich wiederum als Summe der Laufzeiten der in ihnen enthaltenen Teilaufgaben. Für G_1 aus Bild 1 ergibt sich also:

$$L(G_1) = \max \{X_1 + X_2, X_1 + X_4, X_3 + X_4\}$$

Besondere Aufmerksamkeit bei der Modellierung paralleler Programme, aber auch bei der Beschreibung fehlertoleranter Systeme, erlangten die serienparallelen Graphen [12; 14]. Sie erlauben die Darstellung vieler in der Praxis auftretender Probleme, da sie mit Serienkomposition und Parallelschaltung die wichtigsten Konstrukte in der Programmierung paralleler Anwendungen darstellen können. Ihre große Bedeutung verdanken sie allerdings auch der Anwendbarkeit einfacher und effizienter Analysetechniken[3]. Der Graph G_1 aus Bild 1 ist der einfachste nichtserienparallele Graph, der sogenannte N-Graph. Durch Hinzufügen der Kante (T_3, T_2) wird dieser Graph serienparallel (G_2). Verschiedene Autoren entwickelten approximative Analyseverfahren für nicht serienparallele Graphen, die darauf beruhen, durch geschicktes Hinzufügen und Entfernen von Kanten zu einem einfach zu bewertenden

[1] Wir bezeichnen im folgenden solche Graphen auch als Präzedenzgraphen.

[2] Für Gesamtlaufzeit schreiben wir im folgenden auch Bearbeitungszeit bzw. Wert des Graphen.

[3] Eine formale Definition serienparalleler Graphen findet sich z.B. in [12].

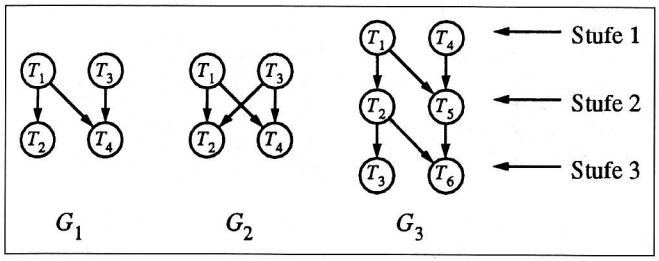


Bild 1: Beispiele für Präzedenzgraphen.

serienparallelen Graphen zu kommen [2; 7].

Der größte Nachteil der serienparallelen Graphen ist sicherlich die Beschränkung der Graphstruktur. Insbesondere ist es nicht möglich, lokale Synchronisationsmechanismen darzustellen, die häufig bei der numerischen Lösung von Gleichungssystemen auftreten (siehe Abschnitt 2.1).

Eine neue Klasse von Graphen, die insbesondere für die Beschreibung von solchen iterativen Strukturen geeignet erscheinen, sind die hier definierten regelmäßigen Graphen. Diese bestehen aus mehreren Stufen, in denen die Teilaufgaben jeweils unabhängig voneinander bearbeitet werden können. Die Abhängigkeiten zwischen den Stufen weisen jeweils die gleiche Struktur auf. G_3 aus Bild 1 ist ein einfacher regelmäßiger Graph mit 3 Stufen und 2 Spalten.

Wir konnten bei der Bewertung von Beispielen aus der Praxis mit Hilfe der Markovanalyse und der Simulation feststellen, daß bei Erweiterung des Präzedenzgraphen um eine Stufe der Mittelwert bzw. die Varianz jeweils um einen konstanten Wert $\Delta\mu$ bzw $\Delta\sigma^2$ zunehmen. Dieses Verhalten, das wir im folgenden als *stationäres* Verhalten bezeichnen, trat meist schon nach wenigen Stufen ein. Somit kann man bei der Bewertung von größeren regelmäßigen Strukturen die Ergebnisse von kleineren Graphen hochrechnen, auch dann, wenn die Bewertung mit herkömmlichen Analyseverfahren wie etwa der Markovanalyse wegen der Zustandsraumexplosion nicht mehr möglich ist.

Im folgenden Kapitel wird die genaue Definition der hier betrachteten Klasse von regelmäßigen Graphen angegeben. Daneben werden einige Beispiele genannt, die zeigen, daß mit dieser Klasse von Graphen Strukturen zu beschreiben sind, die oft im Bereich der parallelen Programmierung angetroffen werden. In Kapitel 3 werden einige Vermutungen für das Verhalten dieser Graphen aufgestellt. Für den einfachsten nichttrivialen Fall eines regelmäßigen Graphen wird außerdem eine Beweisskizze für die vermuteten Eigenschaften angegeben. In Kapitel 4 werden einige Ergebnisse für bestimmte Anwendungen vorgestellt, welche die Vermutungen aus Kapitel 3 stützen.

2 Regelmäßige Graphen

In diesem Kapitel soll die in der Einleitung bereits informell vorgestellte Klasse der regelmäßigen Graphen formal definiert werden. Anschließend wird anhand von Beispielen gezeigt, daß die so definierte Klasse von Graphen häufig bei der Modellierung praktischer Probleme anzutreffen ist.

DEFINITION 1 (regelmäßiger Graph)

Es sei $G = (T, E)$ ein gerichteter, azyklischer Graph mit der Knotenmenge $T = \{T_{ij} \mid i \in \{1, ..., n\}, j \in \{1, ..., m\}\}$ und Kantenmenge $E \subseteq T \times T$. G heißt (n-stufiger) *regelmäßiger* Graph genau dann, wenn es eine Menge W gibt, so daß gilt:

$$W \subseteq \{1, ..., m\} \times \{1, ..., m\}$$

$$E = \{(T_{ij}, T_{i+1\hat{j}}) \mid \forall i \in \{1, ..., n-1\} : (j, \hat{j}) \in W\}$$

$$\forall T_{ij} \in T, i \neq 1 : pred(T_{ij}) \neq \emptyset$$

$$\forall T_{ij} \in T, i \neq n : succ(T_{ij}) \neq \emptyset$$

$pred(T_{ij}) = \{t \in T \mid (t, T_{ij}) \in E\}$ ist die Menge aller Vorgängerknoten von T_{ij}; $succ(T_{ij}) = \{t \in T \mid (T_{ij}, t) \in E\}$ ist die Menge aller Nachfolger von T_{ij}. j soll Spaltenindex, i soll Zeilenindex oder Stufenindex heißen. Die Menge aller Knoten mit gleichem Stufenindex wird als Stufe, die Menge aller Knoten mit gleichem Spaltenindex als Spalte bezeichnet.

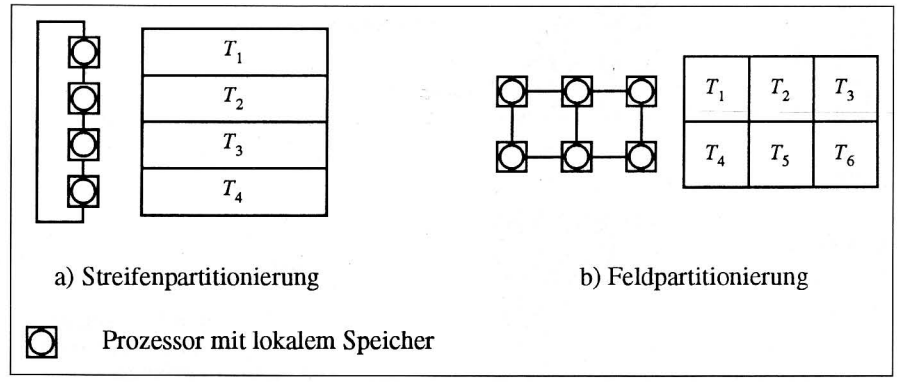
Durch die doppelte Indizierung soll die übersichtliche Darstellung des Graphen in Form eines Feldes verdeutlicht werden. Die Abhängigkeiten zwischen den Stufen wiederholen sich auf jeder Stufe.

Die folgenden Beispiele zeigen, daß die oben definierte Klasse von Graphen praktische Relevanz besitzt.

2.1 Iterationsverfahren

Typischer Vertreter dieser Klasse von Problemen ist die numerische Lösung von Gleichungssystemen bei Verwendung eines Relaxationsverfahrens. Bei Anordnung der Variablen in einem zweidimensionalen Feld bieten sich je nach Zielrechnerarchitektur verschiedene Möglichkeiten an, die Daten auf die Prozessoren zu verteilen. Sind die Prozessoren in einem Ring angeordnet, erscheint die Streifenpartitionierung sinnvoll (Bild 2a), bei einer Feld-

Bild 2: Partitionierungsmöglichkeiten von Variablenfeldern.



konfiguration bietet sich die Feldpartitionierung an (Bild 2b).

Bei der Modellierung solcher iterativer Verfahren mit regelmäßigen Graphen wird jeder Stufe des Graphen ein Iterationsschritt zugeordnet, jeder Prozessor bearbeitet die Teilaufgaben einer Spalte des Graphen. Die Datenabhängigkeiten zwischen zwei Iterationsschritten und werden durch Kanten zwischen den Stufen beschrieben. Wir betrachten im folgenden zwei Fälle:

(1) *Globale Synchronisation:* Hier muß jeder Prozessor auf die Fertigstellung der Bearbeitung aller Teilaufgaben der Stufe i warten, bevor er mit der Bearbeitung seiner Teilaufgabe auf Stufe $i+1$ beginnen kann. Für die Menge W (siehe Definition 1) gilt dann:

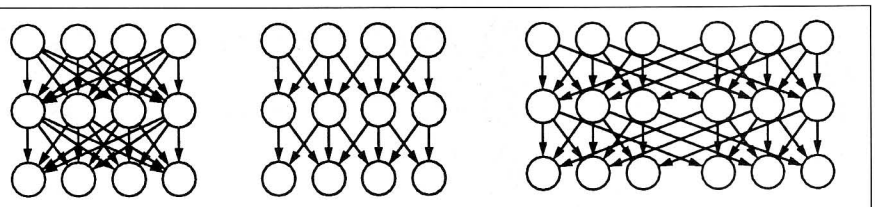
$$W = \{1, ..., m\} \times \{1, ..., m\}$$

Bild 3a zeigt diesen Fall für die Prozessorringkonfiguration.

(2) *Lokale Synchronisation:* In diesem Fall benötigt ein Prozessor nur Werte aus den Bereichen seiner unmittelbaren Nachbarn. Für die Menge W müssen also in Abhängigkeit von der Architektur des Zielrechners zwei Fälle unterschieden werden.

(a) Ringkonfiguration (Bild 3b):

$$W = \{ (j, j) | j \in \{1, ..., m\} \} \cup \{ (j, j-1) | j \in \{2, ..., m\} \} \cup \{ (j, j+1) | j \in \{1, ..., m-1\} \}$$



a) Aufgabenstruktur für globale Synchronisation ($m = 4$)

b) Aufgabenstruktur für Prozessorringkonfiguration mit lokaler Synchronisation ($m = 4$)

c) Aufgabenstruktur für Prozessorfeldkonfiguration mit lokaler Synchronisation ($l = 3$, $b = 2$, $m = 6$) .

Bild 3: Präzedenzgraphen für Feld- und Ringkonfiguration bei verschiedenen Synchronisationsstrukturen.

(b) Feldkonfiguration (Bild 3c)[4]:

$$W = \{(j, j) | j \in \{1, ..., m\}\} \cup \{(j, j+1) | j \bmod l \neq 0\} \cup \{(j, j-1) | j \bmod l \neq 1\} \cup \{(j, j+l) | j \in \{1, ..., (b-1) \cdot l\}\} \cup \{(j, j-l) | j \in \{l+1), ..., l \cdot b\}\}$$

2.2 Makropipelining

Im letzten Abschnitt wurden Verfahren aus dem Bereich der Datenpartitionierung behandelt. Eine andere Möglichkeit, ein Verfahren zu parallelisieren, ist die Algorithmenpartitionierung [1], bei der das Lösungsverfahren in eine Reihe von Teilschritten zerlegt wird, die auf verschiedenen Prozessoren bearbeitet werden. Von Makropipelining spricht man, wenn diese Einheiten so linear angeordnet werden, daß die Ausgabedaten einer Einheit als Eingabedaten für die nächste dienen. Für drei solche Bedieneinheiten ergibt sich der Präzedenzgraph aus Bild 4 (vgl. [11]). Pipelinestrukturen der Breite k und der Tiefe n sind ab Stufe k bis Stufe $(n-k+1)$ regelmäßig im Sinne von Definition 1.

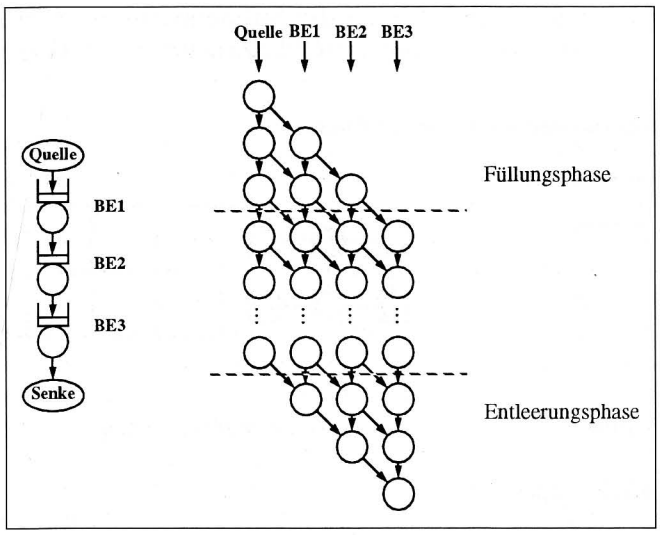


Bild 4: Aufgabenstruktur für Makropipelining.

3 Vermutungen

Bei der Analyse von Beispielstrukturen mit Hilfe der transienten Markovanalyse und der Simulation wurde für die Gesamtbearbeitungszeit ein regelmäßiges Verhalten beobachtet. Die folgenden, aus diesen Beobachtungen resultierenden Vermutungen konnten für den allgemeinen Fall noch nicht bewiesen werden, für den einfachsten nicht serienparallelen Fall folgt im nächsten Abschnitt eine Beweisskizze. Sei $G = (T, E)$ ein n-stufiger regelmäßiger Graph, wobei die Zufallsvariable X_{ij} der Knoten T_{ij} einer Spalte j alle die gleiche Verteilung besitzen. R_i sei die Bearbeitungszeit des i-stufigen Graphen G_i.

(1) Sei $E_i := |E[R_i] - E[R_{i-1}]|$ und $D_i := |Var[R_i] - Var[R_{i-1}]|$ sowie $E[R_0] := Var[R_0] := 0$.[5]

Dann gilt:

$$\forall \varepsilon > 0 : \exists l_1 \in N \Rightarrow \forall i, j > l_1 : |E_i - E_j| < \varepsilon$$
$$\forall i, j > l_2 : |D_i - D_j| < \varepsilon \Rightarrow \forall i, j > l_2 : |D_i - D_j| < \varepsilon$$

(2) Es sei $B_i := R_i - R_{i-1}$ mit Verteilungsfunktion $F_i(t) = P(B_i \leq t)$. Die Verteilungsfunktion F_i konvergiert schwach gegen eine Verteilungsfunktion F, d.h. $\lim_{i \to \infty} F_i = F$.

(3) Es sei R_{ij} die Bearbeitungszeit der j-ten Spalte des Graphen G_i mit Verteilungsfunktion $A_i(t) = P(R_{ij} < t)$. Die Verteilungsfunktion A_i konvergiert schwach gegen eine Normalverteilung.

[4] l ist die Länge und b die Breite des Feldes.

[5] $E[X]$ bezeichnet den Erwartungswert der Zufallsgröße X und $Var[X]$ ihre Varianz.

Vermutung 1 sagt aus, daß ab einer Stufe i bei Erweiterung des Graphen um eine weitere Stufe $i+1$ der Mittelwert und die Varianz der Gesamtbearbeitungszeit des Graphen um einen konstanten Wert $\Delta\mu$ beziehungsweise $\Delta\sigma^2$ zunehmen. Dieses Verhalten tritt beim Graphen aus Bild 3b schon nach wenigen Stufen ein. Wir sagen, daß sich das System ab Stufe i im *stationären* Zustand befindet. Dieses Verhalten vereinfacht die Analyse regelmäßiger Strukturen erheblich. Aufgabenstrukturen müssen nur bis zu einer Tiefe untersucht werden, auf der sich stationäres Verhalten einstellt. Für größere Strukturen lassen sich Mittelwert und Varianz der Gesamtlaufzeit durch Addieren von $\Delta\mu$ und $\Delta\sigma^2$ errechnen. Man kann das stationäre Verhalten als eingetreten betrachten, wenn die Differenz der Zuwächse für Varianz und Mittelwert kleiner wird als eine Schranke ε.

3.1 Beweis für den einfachsten, nichtserienparallelen Fall

Der einfachste, nichttriviale Fall eines regelmäßigen Graphen ist die stufenweise Wiederholung des N-Graphen G_1 aus Bild 1. Für diesen Graphen lassen sich die Vermutungen 1–3 beweisen. Die unten angegebene Beweisskizze folgt im wesentlichen den Notizen in [3]. Die Beweisidee besteht darin, daß sich die Bearbeitungszeit eines n-stufigen Graphen dieser Art darstellen läßt als Summe der Wartezeiten des n-ten Kunden im GI/G/1-System plus einer Summe von n unabhängigen Zufallsgrößen.

Satz:

Es sei G_n ein n-stufiger regelmäßiger Graph mit Knotenmenge

$T = \{T_{ij} \mid i \in \{1, ..., n\}, j \in \{1, 2\}\}$ und
$W = \{(1, 1), (2, 2), (1, 2)\}$

dann gelten die Vermutungen 1–3.

Beweisskizze:

zu 3) X_{ij} sei die Bearbeitungszeit des Knoten T_{ij} im Graphen G_n. W_i bezeichne die Zeit zwischen der Fertigstellung des Knotens T_{i1} und dem Beginn des Knotens $T_{i+1,2}$. Es gilt dann:

$$R_{i2} = W_{i-1} + \sum_{k=1}^{i-1} X_{k1} + X_{i2}$$

$$R_{i1} = \sum_{k=1}^{i} X_{k1}$$

Für die Größe W_i gilt:

$W_1 = \max\{0, X_{12} - X_{11}\}$

$W_i = \max\{0, W_{i-1} + X_{i2} - X_{i1}\}$

Das ist aber gerade die Rekursionsbeziehung für die Wartezeit des i-ten Kunden im Wartesystem GI/G/1, falls man setzt:
X_{i2} := Bedienzeit des $(i-1)$-ten Kunden
X_{i1} := Zwischenankunftszeit zwischen dem $(i-1)$-ten und dem i-ten Kunden

Die Wartezeit im GI/G/1-System strebt für $E[X_{i1}] > E[X_{i2}]$ bekanntlich in einen Gleichgewichtszustand (siehe z.B. [8]). Daraus folgt, daß die Summe $\sum X_{k1}$ für wachsendes k dominiert und die Spaltenbearbeitungszeiten R_{i1} bzw. R_{i2} folglich asymptotisch normalverteilt sind. Für $E[X_{i1}] < E[X_{i2}]$ hat die Kante $(T_{i1}, T_{i+1,2})$ mit wachsender Stufenzahl keinen Einfluß mehr, weshalb die Spaltenbearbeitungszeiten auch in diesem Fall asymptotisch normalverteilt sind.

zu 2) Analoge Folgerungen sind für die Gesamtbearbeitungszeit $R_i = \max\{R_{i1}, R_{i2}\}$ des Graphen möglich. Für den einfachsten, nichttrivialen regelmäßigen Graphen sind demnach sowohl die Spaltenbearbeitungszeiten, als auch die Gesamtbearbeitungszeit asymptotisch normalverteilt.

zu 1) Für Aussagen über die Fertigstellungszeit einer Stufe können ebenfalls bekannte Ergebnisse für GI/G/1-Systeme herangezogen werden. Aus Ergebnissen von [9] für die Zwischenabgangszeiten im System GI/G/1 erhalten wir für die Fertigstellungszeiten folgende Zusammenhänge:

$$B_{i+1,2} = R_{i+1,2} - R_{i2} = X_{i+1,2} - \min\{0, W_{i-1} + X_{i2} - X_{i1}\}$$

$$B_{i+1,1} = R_{i+1,1} - R_{i1} = X_{i+1,1}$$

Für den Fall $W_{i-1} + X_{i2} - X_{i1} < 0$ ist die Verteilungsfunktion von $B_{i+1,2}$ die Faltung der Bedienzeitverteilungsfunktion $X_{i+1,2}$ mit der Idlezeitverteilung ($W_i(x)$ für $x < 0$) im GI/G/1-System. Für den anderen Fall $W_{i-1} + X_{i2} - X_{i1} \geq 0$ gilt $B_{i+1,2} = X_{i+1,2}$. Da die Verteilung der Idlezeit im System GI/G/1 bekanntlich eine Grenzverteilung besitzt, ist auch die Konvergenz der Verteilungsfunktionen und der Momente der Fertigstellungszeit der einzelnen Stufen im betrachteten regelmäßigen Graphen sichergestellt.

Es gibt weitere Beispiele regelmäßiger Graphen mit größerer Breite, die ebenfalls nach obigem Schema behandelt werden können. Für den allgemeinen Fall eines regelmäßigen Graphen ist uns bisher allerdings kein Beweis für die Vermutungen 1–3 gelungen.

4 Ergebnisse

In diesem Abschnitt werden einige Ergebnisse für die Beispiele aus Abschnitt 2.1 präsentiert, welche die Vermutungen aus Abschnitt 3 belegen sollen. In [4] sind weitere Beispiele angegeben.

4.1 Prozessorringkonfiguration

Tabelle 1 zeigt den Mittelwert der Gesamtlaufzeit für die Prozessorringkonfiguration mit lokaler Synchronisation, wobei über die Stufenzahl und die Breite des Graphen variiert wurde. Die Ergebnisse wurden mit Hilfe der transienten Markovanalyse gewonnen.

In der Horizontalen ist die Variation der Spalten der Graphen aufgetragen und in der Vertikalen die Variation der Stufenzahl der Graphen. Die Differenzen der Werte der Stufen 7 und 8 bzw. 8 und 9 zeigen, daß bei dieser Stufenzahl bereits das stationäre Verhalten (im numerischen Sinne) der Graphen erreicht ist. Bei der globalen Synchronisation ergibt sich die Gesamtlaufzeit als Summe identisch verteilter, stochastisch unabhängiger Zufallsvariable. Daher ist der Zuwachs des Erwartungswertes natürlich immer konstant und es wird für jede Breite nur ein Wert angegeben. Auch für die Varianz der Gesamtlaufzeit tritt bereits nach 9 Stufen das stationäre Ver-

Tabelle 1: Ringkonfiguration, ERL (2, 1), Mittelwert.

Erwartungswert der Gesamtbearbeitungszeit				
Breite	3	4	5	6
Tiefe (lokal)				
6	18.5116	19.7378	20.5922	21.2394
7	21.5735	22.9759	23.9439	24.6704
8	24.6354	26.2140	27.2958	28.1016
9	27.6973	29.4521	30.6476	31.5330
8–7	3.0619	3.2381	3.3519	3.4312
9–8	3.0619	3.2381	3.3518	3.4314
global	3.2130	3.5472	3.8083	4.0223
global-lokal	0.1511	0.3091	0.4564	0.5909
%	4.7	8.7	12.0	14.7

halten ein, wie die Ergebnisse in [4] zeigen. Dort finden sich auch weitere Beispiele, in denen unter anderem den einzelnen Spalten auch unterschiedliche Verteilungen zugeordnet wurden.

4.2 Makropipelining

Tabelle 2 zeigt die erwarteten Fertigstellungszeitpunkte und die zugehörigen Varianzen zu dem Beispiel aus Abschnitt 2.2, wobei wir von einer Pipeline mit zwei Bedieneinheiten und folgenden Verteilungsannahmen für die Bedienzeiten und den Ankunftsprozeß ausgehen[6]:

$\forall i \in \langle 1, n \rangle : X_{i1} \overset{v}{\sim} ERL\ (2, 1),$
$X_{i2} \overset{v}{\sim} EXP\ (1), X_{i3} \overset{v}{\sim} ERL\ (2, 2)$

Die Tabelle zeigt, daß auch dieser schließlich regelmäßige Graph einen stationären Zustand erreicht. Die Zuwächse für Mittelwert und Varianz unterscheiden sich ab Stufe 14 kaum noch. Damit kann man auch bei diesem Graphen von einem stationären Verhalten sprechen.

5 Ausblick

Die hier eingeführten regelmäßigen Graphen bilden eine interessante Teilmenge der Menge der gerichteten, azyklischen Graphen. Zum einen führt die Modellierung vieler praktischer Probleme auf solche regelmäßige Strukturen, zum anderen wurde in allen analysierten Beispielen das Auftreten eines stationären Verhaltens für den Erwartungswert und die Varianz der Gesamtbearbeitungszeit der Graphen festgestellt. Aufgrund dieser Eigenschaft können Mittelwert und Varianz kleiner regelmäßiger Graphen zum Beispiel mit Hilfe der Markovanalyse bestimmt werden, und der Mittelwert bzw. die Varianz großer regelmäßiger Graphen lassen sich unter Ausnutzung des stationären Verhaltens ermitteln. Für den Spezialfall des einfachsten nichtserienparallelen Graphen konnte eine Beweisskizze für die angegebenen Vermutungen über das stationäre Verhalten dieser Graphen angegeben werden. Für den allgemeinen Fall fehlt dieser Beweis noch.

Die Bewertung der Makropipelinestruktur zeigt, daß die transiente Phase der Gesamtlaufzeit (das sind die Stufen, bis sich das stationäre Verhalten einstellt) in Abhängigkeit von der Aufgabenstruktur sehr unterschiedlich sein kann. Während sich bei den Graphmodellen aus Bild 3a und b das stationäre Verhalten bereits nach wenigen Stufen einstellt (im Fall der lokalen Synchronisation etwa ab Stufenzahl = Spaltenzahl), tritt dies bei der Pipelinestruktur erst sehr viel später ein (im Fall der Pipelinestruktur aus Bild 4 erst ab einer Stufenzahl von ungefähr 80). Prinzipiell kann man feststellen, daß bei größeren Abhängigkeiten zwischen den Stufen der stationäre Zustand schneller erreicht wird.

Auch bei Variation der Verteilungsparameter ist die Hochrechnung aus bereits berechneten Ergebnissen möglich. Falls man nämlich statt der Zufallsvariablen X_{ij} linear transformierte Größen $X_{ij}^{(\text{trans})} = cX_{ij} + \theta$, mit $c, \theta \in R$ für die Knotenlaufzeiten verwendet, erhält man für die Gesamtlaufzeit $X_G^{(\text{trans})} = n\theta + cX_G$. Dabei ist X_G die Laufzeit des Graphen mit den nicht transformierten Zufallsvariablen. Dies bedeutet, daß sich stochastische und deterministische Zeiten sehr gut kombinieren lassen.

Die Definition der *„Regelmäßigkeit"* kann in verschiedene Richtungen erweitert werden. In der hier verwendeten Definition werden die Knoten des Graphen durch ihre „Koordinaten" (i, j) unterschieden. Es sind auch Definitionen denkbar, bei denen Knoten durch mehr als zwei Indizes charakterisiert werden. Für Koordinaten (i, j, l) kann dies, im Gegensatz zur Feldanordnung bei doppelter Indizierung, als räumliche Anordnung der Knoten interpretiert werden. Eine andere Erweiterung ergibt sich, wenn Abhängigkeiten zwischen den Knoten nicht nur über eine Stufe reichen, sondern Abhängigkeiten über mehrere Stufen erlaubt sind.

Literatur

[1] *Agrawal, D., Jain, R.:* A Pipelined Pseudoparallel System Architecture for Real-Time Dynamic Scene Analysis. IEEE Transactions on Computers, C-31 (10), (1982), S. 952–962.

[2] *Dodin, B.:* Bounding the Project Completion Time Distribution in PERT Networks. Operations Research Vol. 33, S. 862–880, 1985.

[3] *Ferschl, F.:* Seminar für angewandte Stochastik an der Universität München. Private Mitteilung, Brief vom 25. Oktober 1989.

Tabelle 2: Pipelinestruktur.

MP	1	2	3	4	5	14	15	19	20
$E[X_G]$	4.00	6.44	8.67	10.81	12.90	31.13	33.14	41.15	43.15
$Var[X_G]$	3.50	5.22	6.88	8.56	10.28	26.93	28.87	36.70	38.67

[6] $X \overset{v}{\sim} F$ heißt, daß der Zufallsgröße X die Verteilung F zugeordnet ist.

[4] *Fleischmann, G.:* Leistungsbewertung paralleler Programme für MIMD-Architekturen: Modellbildung und mathematische Analyse. Dissertation, Universität Erlangen-Nürnberg, 1990.

[5] *Fleischmann, G., Gente, M., Hofmann, F., Bolch, G.:* Performance Analysis of Parallel Programs Based on Model Calculations; In: Parallel Computing, Vol. 20, No. 10–11, Nov. 1994, S. 1 583–1 603.

[6] *Gelenbe, E.:* Multiprocessor Performance. John Wiley & Sons, New York, 1989.

[7] *Hartleb, F., Mertsiotakis, V.:* Bounds for the Mean Runtime of Parallel Programs. In Proceedings fo the Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation '92, (1992), S. 197–210.

[8] *Kleinrock, L.:* Queueing systems I. John Wiley & Sons, New York, 1975.

[9] *Litko, J. R.:* GI/G/1 Interdeparture Time and Queue-Length Distributions via the Lagurerre Transform. Queueing Systems, 4, (1989), S. 367–382.

[10] *Madala S., Sinclair, J. B.:* Performance of Synchronous Parallel Algorithms with Regular Structures. IEEE Transactions on Parallel and Distributed Systems, 2 (1), 1991.

[11] *Mohan, J.:* Performance of Parallel Programs: Model and Analyses. PhD thesis, Carnegie Mellon University, 1984.

[12] *Sahner, R. A., Trivedi, K. S.:* SPADE: A Tool For Performance and Reliability Evaluation. In: N. Abu El Ata (Hrsg.): Modelling Techniques and Tools For Performance Analysis, S. 147–163, 1986.

[13] *Sötz, F.:* A Method for Performance Prediction of Parallel Programs. In: H. Burkhart (Hrsg.): CONPAR 90 – VAPP IV, 1990.

[14] *Sötz, F., Werner, G.:* Lastmodellierung mit stochastischen Graphen zur Verbesserung paralleler Programme auf Multiprozessoren mit Fallstudie. In: P. Müller-Stoy (Hrsg.): Architektur von Rechensystemen, 1990.